Motion Correction of Lung CT Images With An Application To Oncology

by

Tony Thompson
under the supervision of:

Prof. Ke Chen

UNIVERSITY OF
LIVERPOOL

Thesis submitted in accordance with the requirements
of the University of Liverpool for the
degree of Doctor in Philosophy.

February 2019

Contents

Acknowledgment

I would like to express my gratitude to my PhD supervisor Prof Ke Chen, along with my supervisors at the CCC Dr John Fenwick and Dr Colin Baker, for their support and guidance throughout my doctoral studies at the University of Liverpool. I would also like to thank Simon Temple, Louise Turtle and Konstantin Lavrenkov at the CCC for their help and patience with the medical aspects of my work.

In addition, I would like to thank Prof Natalia Movchan and Dr David Lewis for their advice and constructive criticism of my work. Moreover, I would like to thank my current and former colleagues within CMIT including Dr Bryan Williams, Dr Jack Spencer, Dr Mazlinda Ibrahim, Mike Roberts, Daoping Zhang and Abdul Jumaat for all the helpful and insightful discussions throughout this time.

Additionally, I would like to give a huge thank you to my family and friends, and especially my parents Julie and Mark Green, for all their encouragement and support they gave me throughout my studies.

Finally, I would like to thank the EPSRC body and CCC for my financial support.

Abstract

The aim of image registration is to align sets of similar images which have been captured at different time points, from different perspectives or obtained using different imaging modalities (e.g. CT, MRI, X-ray). In oncology, image registration is a very powerful tool which can be used in an array of different applications such as anatomic image segmentation, 4D dose accumulation maps and lung ventilation maps. In order to align a given set of images, we first assign one image in the set to be the 'fixed' reference image to which we align the remaining 'moving' template images. We are then tasked with finding suitable transformations which deform the template images to match the reference image. In this thesis, we model the image registration problem mathematically through the minimisation of an energy functional.

We begin by proposing an improved non-linear multigrid method, based upon the method proposed by Chumchob and Chen in [33], via a more accurate analysis of the scheme and new solver to improve convergence, accuracy and CPU time. Next we extend our improved Chumchob-Chen model to incorporate an additional constraint to prevent folding in the transformation, thus leading to physically accurate diffeomorphic registrations. After this we further extend our proposed constrained model to improve robustness with regard to accuracy in cases of severe folding, in addition to parameter choice. We then demonstrate these improvements using a combination of real lung CT images and a synthetic hand X-ray image set.

Next we consider a different approach to addressing the problem of folding in the transformation, by formulating an inverse consistent image registration model based upon the model first proposed by Christensen and Johnson in [28]. Our proposed idea is to linearise the inverse consistency constraint in the Christensen-Johnson model, which is extremely expensive to compute with regard to computational cost due to its non-linear nature, in addition to implementing a fast non-linear multigrid scheme to further help reduce the computational cost of the model. We then perform some numerical tests on a mix of real CT images and a synthetic example, to highlight the advantages of our proposed inverse consistent model.

Finally, we present three 3D image registration models based upon the models discussed throughout this thesis, in addition to 3D extensions of the associated non-linear multi-

grid schemes. We then show some preliminary results, using eight examples taken from the Hugo image database [80] along with clinically drawn contours for nine different objects within the CT images, comparing our proposed models with a state of the art commercial software used in hospitals.

Publications and presentations

Publications

- T. Thompson and K. Chen. *A more robust multigrid for diffusion type models*, Journal of Computational and Applied Mathematics, 2019
- T. Thompson and K. Chen. *An effective diffeomorphic model and its fast multigrid algorithm for registration of lung CT images*, Computational Methods for Applied Mathematics, 2019
- T. Thompson and K. Chen and J. Fenwick, *Validation of two non-folding 3D registration models for use in oncology, In preperation*

Presentations

- T. Thompson and K. Chen and C. Baker and J. Fenwick, *Non-linear optical flow for lung CT registration*, SIAM Conference on Imaging Science (IS16), Albuquerque (USA), May 23rd-26th, 2016
- T. Thompson and K. Chen and C. Baker and J. Fenwick, *Non-linear optical flow with multigrid acceleration for medical image registration*, Alan Tayler Day, Oxford (UK), October 28th, 2016

List of figures

List of tables

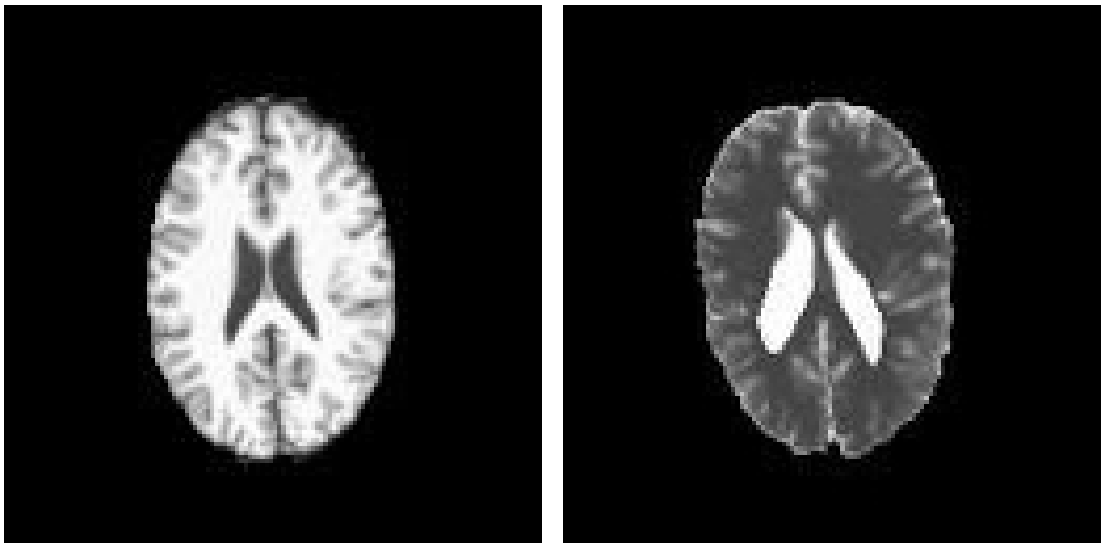
Chapter 1

Introduction

One of the most useful, and challenging, tasks in image processing is image registration. The goal of image registration techniques is to align pairs or sequences of similar images by finding correspondences between various features within the images. Other names for image registration include image fusion, image matching and image warping. One area where registration plays a crucial role is in medical imaging, especially in problems which involve images of the lungs. The reason why image registration is especially important in such medical image problems is due to the various types of motion which can occur when treating patients with lung diseases over long periods of time. Some examples of the different types of motion include the tracking of tumour movement over time in response to treatment, and movement resulting from patient breathing during the scanning process. Therefore image registration is required for tasks such as radiation therapy [136], determining changes in lung volume and function [20], radiation dose estimation [38, 64, 102] and motion correction [107, 119].

One specific example of where image registration can be used in the treatment of patients with lung cancer, is deforming contours between image phases. Typically a radiologist will contour several key areas on one phase of the 4D image sequence including the gross tumour volume (GTV), clinical target volume (CTV) and planning target volume (PTV). The GTV is simply an outline of the visible tumour on all slices of the image, the CTV is an expanded region encompassing the extreme boundaries of the tumour motion across all image phases and the PTV is a further expanded region to include an area of uncertainty to which radiation will be applied. Due to the time consuming process of contouring the tumour on a single phase, possibly taking up to several hours, image registration can be used to track the motion of the tumour across the other phases to give the CTV region. From this the PTV region can be computed easily, and the treatment can therefore be commenced.

In this thesis, we aim to develop mathematical registration models with a focus on physical deformations. The remainder of this chapter consists firstly of a brief introduction



(a) Reference image R. (b) Template image T.

Figure 1.1: Example of multi-modal images consisting of a pair of differently weighted brain MRI images.

into how image registration is formulated mathematically before giving an outline of this thesis.

1.1 Introduction to image registration

The process of image registration involves searching for a geometric transformation between a template (or target) image and a reference (or source) image, with the goal of aligning the two images by applying the found transformation to the template image. In order for us to be able to match the two images, it is typical to assume the template and reference images are of the same visual scene with some variations.

We can classify image registration into two categories, namely mono-modal and multi-modal image registration. In the former class the set of images to be registered are obtained using the same imaging apparatus, and so we can match the images according to intensity values since the same features have the same values in each image. Now for the latter case, we are unable to use intensity values to match the images since different features possess different values as illustrated in Figure 1.1. However, the main focus of this thesis will be on the former type of image regisration, i.e. mono-modal registration. For the case of mono-modal images, where we employ intensity based registration methods, we can further classify such methods into different categories. Some examples of these categories include physical (rigid and non-rigid), mathematical (linear and non-linear) and complexity (parametric and non-parametric). Let us consider the complexity categories, i.e. parametric and non-parametric registration. In the former

(a) Reference image R. (b) Template image T.

Figure 1.2: Two squares example.

class, the transformation which we seek is governed by a small, finite number of parameters. An example of this is affine registration (which allows for translations, rotations, scaling and shearing) where we are required to determine 6 parameters in the 2D case, and 12 parameters in the 3D case. However, the focus of this thesis will be on the second class of registration models, namely non-parametric registration. For this class of registration model, we typically determine the desired transformation by forming an energy functional and look to find its minima. The functional in question is usually obtained based on some physical process, for example elasticity or fluid flow.

The challenge in image registration arises from the fact the associated minimisation problem which needs to be solved is ill-posed in the sense of Hadamard since the solution is not unique. Supposing we are given the reference and template images as shown in Figure 1.2. We can clearly see the transformation describing a pure translation as well as the transformation combining a translation with a clockwise rotation through $\frac{\pi}{2}$ both match the template image in Figure 1.2(b) to the reference image in Figure 1.2(a), thus we see the solution is not unique unless some additional information is given. We discuss how the problem of ill-posedness can be overcome in §2.3. Non-parametric models are an ideal choice for real-life problems involving non-uniform deformations, one such example being motion correction of lung images where parametric models perform poorly. However, these models are not without drawbacks which need to be addressed. One of the biggest challenges facing non-parametric image registration is computational speed.

Typically the transformation is determined pixel by pixel (or voxel by voxel in 3D), and so the number of unknowns which need to be computed is proportional to the

number of pixels (or voxels) in the images. Therefore we see the numerical solution the non-parametric model may be practically too slow, especially for large images. Thus we must look to construct computationally robust models and numerical schemes for real-life applications.

1.2 Outline of thesis

The remainder of this thesis will be set out as followed.

Chapter 2 – Mathematical preliminaries

In this chapter we introduce some basic mathematical tools which we use throughout the remainder of this thesis. We introduce some useful definitions and theorems, in addition to including examples of relevant topics which include normed linear spaces, functions of bounded variation, calculus of variations, ill-posed inverse problems and regularisation, discretisation of partial differential equations (PDEs) using finite difference methods (FDMs), the iterative solution of both linear and non-linear systems of equations and a brief introduction into multigrid (MG) methods.

Chapter 3 – Mathematical models for image registration

In this chapter we discuss how the image registration problem can be formulated in a mathematical setting. We do this by introducing the concept of similarity measures such as the sum of squared distances (SSD) and mutual information (MI) measures. Next we outline the mathematical formulation of the parametric and non-parametric registration models. For the former type we discuss the rigid body and affine registration models, while for the latter type we discuss the different possible choices of regularisation such as elastic and curvature based regularisation terms. Finally we introduce the discretise-optimise and optimise-discretise schemes for solving the registration problem.

Chapter 4 – A more robust multigrid for diffusion type registration models

In this chapter we introduce a robust non-linear multigrid (NMG) method applied to a diffusion model based upon the scheme proposed by Chumchob and Chen in [33]. We include a more accurate analysis of the proposed smoother scheme using local Fourier analysis (LFA) to achieve optimal efficiency in the NMG scheme, and improve upon the convergence when compared with the Chumchob-Chen model. In addition we propose an extension to the diffusion model to produce transformations which are physical and contain no folding, in addition to robustness with respect to the choice of weighting parameter α via the inclusion of a constraint and adaptive updating of the parameter α. Finally we present some numerical results demonstrating the robustness of our model with respect to folding and parameter choice, in addition to showing how the proposed NMG scheme possesses optimal efficiency. Moreover, we show how our improved analy-

sis of the Chumchob-Chen NMG scheme improves upon the convergence and CPU time in addition to improving the accuracy of the registration.

Chapter 5 – An effective diffeomorphic model and its fast multigrid algorithm for the registration of lung CT images

In this chapter we propose a simplification of the inverse consistent model proposed by Christensen and Johnson in [28] with the goal of producing diffeomorphic transformations (i.e. transformations with no folding) and reducing the computational cost of solving the model numerically. We then describe a NMG scheme, along with three potential smoother schemes, before performing a detailed analysis using LFA to achieve optimal efficiency. Finally we show some numerical tests on real lung CT images to demonstrate how the proposed inverse consistent model achieves good accuracy, while maintaining physically accurate registrations, when compared with a typical diffusion model.

Chapter 6 – Preliminary validation of two non-folding 3D registration models for use in oncology

In this chapter we extend the three proposed models from Chapters 4 and 5 into 3D, in addition to extending the associated multigrid schemes into 3D. After this we perform an analysis on the multigrid schemes in 3D using LFA in order to obtain a multigrid scheme with optimal efficiency. Finally we present some test results comparing the accuracy of our two non-folding registration models with a standard 3D diffusion model on eight real lung CT examples taken from the Hugo dataset [80], before providing some preliminary results comparing the three discussed models with a state of the art commercial software called Eclipse which is currently used in hospitals.

Chapter 7 – Conclusions and future research

In this final chapter, we present some conclusions regarding the work discussed in this thesis in addition to describing some possible avenues of future work related to the described work.

Chapter 2

Mathematical preliminaries

In this chapter, we begin by discussing various basic mathematical concepts which will be used throughout the remainder of this thesis. Then we introduce some relevant theory regarding calculus of variations, before moving on to discuss inverse problems and regularisation. Next we introduce the concept of the discretisation of partial differential equations (PDEs), in addition to some relevant notations. After this we look at some iterative methods used to solve linear systems of equations, before moving onto iterative methods which can be used to solve non-linear equations. Finally we conclude with a brief introduction into multigrid methods and theory.

2.1 Normed linear spaces

Definition 2.1.1 (Linear vector space)**.** *Suppose V is a set where the two operations addition and scalar multiplication are defined. Also suppose that $\boldsymbol{u}, \boldsymbol{v} \in V$ are elements of the set V, with the sum of $\boldsymbol{u}$ and $\boldsymbol{v}$ denoted by $\boldsymbol{u} + \boldsymbol{v}$ and the scalar multiplication of $\boldsymbol{u}$ with an element $\lambda \in F$ of a scalar field F denoted by $\lambda\boldsymbol{u}$. Then we call V a linear vector space, over a scalar field F, if the following ten axioms hold $\forall\, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V$ and $\forall\, \lambda, \mu \in F$:*

(i) **Closure under addition***:*

$$\boldsymbol{u} + \boldsymbol{v} \in V;$$

(ii) **Commutativity under addition***:*

$$\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u};$$

(iii) **Associativity under addition***:*

$$\big(\boldsymbol{u} + \boldsymbol{v}\big) + \boldsymbol{w} = \boldsymbol{u} + \big(\boldsymbol{v} + \boldsymbol{w}\big);$$

(iv) **Identity element of addition**:

$$\exists\, \mathbf{0} \in V \text{ such that } \boldsymbol{u} + \mathbf{0} = \boldsymbol{u};$$

(v) **Existence of additive inverse**:

$$\forall\, \boldsymbol{u} \in V, \exists - \boldsymbol{u} \in V \text{ such that } \boldsymbol{u} + \big(-\boldsymbol{u}\big) = \mathbf{0};$$

(vi) **Closure under scalar multiplication**:

$$\lambda \boldsymbol{u} \in V;$$

(vii) **Distributivity under scalar multiplication (I)**:

$$\lambda\big(\boldsymbol{u} + \boldsymbol{v}\big) = \lambda \boldsymbol{u} + \lambda \boldsymbol{v};$$

(viii) **Distributivity under scalar multiplication (II)**:

$$\big(\lambda + \mu\big)\boldsymbol{u} = \lambda \boldsymbol{u} + \mu \boldsymbol{u};$$

(ix) **Associativity under scalar multiplication**:

$$\lambda\big(\mu \boldsymbol{u}\big) = \big(\lambda\mu\big)\boldsymbol{u};$$

(x) **Identity element of scalar multiplication**:

$$\exists\, I \in F \text{ such that } I\boldsymbol{u} = \boldsymbol{u}.$$

Example 2.1.2 (Linear vector space)**.** *Examples of linear spaces include:*

(i) *The vector space $F[x]$ given by polynomial functions $f(x) = \lambda_0 + \lambda_1 x + \lambda_2 x^2 + \ldots$;*

(ii) *The spaces $\mathbb{R}^n$ and $\mathbb{C}^n\, \forall\, n \in \mathbb{N}$.*

Definition 2.1.3 (Norm and semi-norm)**.** *Let V be a linear vector space over a scalar field F, also let $\boldsymbol{u}$, $\boldsymbol{v} \in V$ be elements of V and $\lambda \in F$ be a scalar. Then a norm $|\cdot|$ on V is a non-negative real valued function $|\cdot| \colon V \to \mathbb{R}$ such that the following properties hold:*

(i) $|\boldsymbol{u}| > 0$ *if* $\boldsymbol{u} > 0$ *and* $|\boldsymbol{u}| = 0 \iff \boldsymbol{u} = 0$;

(ii) $|\lambda \boldsymbol{u}| = |\lambda|\,|\boldsymbol{u}|$;

(iii) $|\boldsymbol{u} + \boldsymbol{v}| \leq |\boldsymbol{u}| + |\boldsymbol{v}|$.

A semi-norm is defined in a similar way to a norm, with the exception property (i) is replaced with

$$|\boldsymbol{u}| \geq 0\, \forall \boldsymbol{u}.$$

Definition 2.1.4 (Normed linear space). *A normed linear space is a linear vector space V which is equipped with a norm $|\cdot|$.*

Example 2.1.5 (p-norm). *Suppose $\boldsymbol{x} \in \mathbb{R}^n$, then for any $1 \leq p \in \mathbb{R}$ the p-norm of $\boldsymbol{x}$ is defined as*

$$|\boldsymbol{x}|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}.$$

In the case when $p = 2$, we recover the Euclidean norm which is defined by

$$|\boldsymbol{x}|_{\mathbb{R}^n} = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

The Euclidean scalar product, denoted by $\boldsymbol{x} \cdot \boldsymbol{y}$, is defined by

$$\boldsymbol{x} \cdot \boldsymbol{y} = |\boldsymbol{x}||\boldsymbol{y}| \cos\theta$$

where θ denotes the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$.

Example 2.1.6 (L_p-norm). *Let f be a function defined on a domain Ω and $1 \leq p \leq \infty$, then the L_p-norm of f on Ω is defined as*

$$|f(x)|_p = \left(\int_\Omega |f(x)|^p \, dx \right)^{\frac{1}{p}}.$$

Note this is a generalisation of Example 2.1.5 since the number of components is now arbitrary.

Example 2.1.7 (L_∞-norm). *When $p = \infty$, we have a special case of the L_p-norm from Example 2.1.6 (namely the L_∞-norm), defined by*

$$|f(x)|_\infty = \sup_x |f(x)|.$$

Definition 2.1.8 (Inner product). *An inner product on a linear vector space V, defined over the scalar field F, is a function $\langle \cdot, \cdot \rangle_V$ on $V \times V$ which satisfies the following conditions:*

(i) **Conjugate symmetry***:*

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_V = \overline{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}_V \, \forall \, \boldsymbol{u}, \boldsymbol{v} \in V;$$

(ii) **Linearity under scalar multiplication***:*

$$\langle \lambda \boldsymbol{u}, \boldsymbol{v} \rangle_V = \lambda \langle \boldsymbol{u}, \boldsymbol{v} \rangle_V \, \forall \, \boldsymbol{u}, \boldsymbol{v} \in V \text{ and } \lambda \in F;$$

(iii) **Linearity under vector addition***:*

$$\langle \boldsymbol{u} + \boldsymbol{v}, \boldsymbol{w} \rangle_V = \langle \boldsymbol{u}, \boldsymbol{w} \rangle_V + \langle \boldsymbol{v}, \boldsymbol{w} \rangle_V \, \forall \, \boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w} \in V;$$

(iv) **Positive definiteness***:*

$$\langle \boldsymbol{u}, \boldsymbol{u} \rangle_V > 0 \, \forall \, \boldsymbol{u} \neq 0 \in V.$$

Example 2.1.9 (Inner product)**.** *The standard example of an inner product is the function* $\langle \cdot, \cdot \rangle_{\mathbb{R}^n}$ *defined on* $\mathbb{R}^n \times \mathbb{R}^n$ *by*

$$\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\mathbb{R}^n} = \boldsymbol{y}^T \boldsymbol{x} = \sum_{i=1}^{n} x_i y_i \; \forall \, \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

Definition 2.1.10 (Support of a function)**.** *Suppose* f *is a real (or complex) valued function whose domain* Ω *is a non-empty set in* $\mathbb{R}^n$ *(or* $\mathbb{C}^n$*). Then the support of* f *is defined as*

$$\sup(f) = \big\{ x \in \Omega \colon f(x) \neq 0 \big\}.$$

Moreover, if $\sup(f)$ *is a compact set (i.e. a closed and bounded set) in* Ω *then we say* $f \in \Omega \subset \mathbb{R}^n$ *has compact support in* Ω*.*

Definition 2.1.11 (Cauchy sequence)**.** *A sequence* $\{x_k\}_{k \in \mathbb{N}}$ *in a normed linear vector space* V *is called a Cauchy sequence if* $\forall \, \varepsilon > 0$ *there exists* K *such that any*

$$k, \, l \geq K \implies |u_k - u_l| < \varepsilon.$$

Furthermore, if every Cauchy sequence converges then we say V *is complete.*

Definition 2.1.12 (Banach space)**.** *A complete normed linear vector space is called a Banach space.*

Definition 2.1.13 (Hilbert space)**.** *An inner product space which is complete with respect to the norm induced by the inner product is called a Hilbert space.*

Example 2.1.14 (Hilbert space)**.** *Two relevant examples of Hilbert spaces are the space* $\mathbb{R}^n$ *together with the Euclidean inner product and the space* $L^2(\Omega)$ *with the inner product*

$$\langle f, g \rangle_{L^2(\Omega)} = \int_{\Omega} f(\boldsymbol{x}) g(\boldsymbol{x}) \, d\Omega.$$

Definition 2.1.15 (Linear operator)**.** *Let* V *and* W *be vector spaces, then a mapping* $\mathcal{L} \colon V \to W$ *is called linear if*

$$\mathcal{L}\big(\lambda_1 \boldsymbol{u}_1 + \lambda_2 \boldsymbol{u}_2\big) = \lambda_1 \mathcal{L}\big(\boldsymbol{u}_1\big) + \lambda_2 \mathcal{L}\big(\boldsymbol{u}_2\big)$$

$\forall \, \boldsymbol{u}_1, \, \boldsymbol{u}_2 \in V$ *and where* $\lambda_1, \, \lambda_2 \in \mathbb{R}$ *are scalars.*

Example 2.1.16 (Linear operator)**.** *A linear operator mapping* $\mathbb{R}^n \to \mathbb{R}^m$ *is defined by a matrix* $\boldsymbol{\mathcal{L}}$ *of size* $m \times n$*. Then given* $\boldsymbol{x} \in \mathbb{R}^n$ *we have*

$$\boldsymbol{y} = \boldsymbol{\mathcal{L}} \boldsymbol{x} \in \mathbb{R}^m.$$

Definition 2.1.17 (Convex set)**.** *A set* $\mathcal{S}$ *is said to be convex if* $\forall \, u, v \in \mathcal{S}$*, and for any*

$0 \leq \lambda \leq 1$, *we have*

$$\lambda u + (1-\lambda)v \in \mathcal{S}.$$

Definition 2.1.18 (Convex function)**.** *A function f, defined on a convex set $\mathcal{S}$, is called convex if $\forall\, u, v \in \mathcal{S}$ and $0 \leq \lambda \leq 1$*

$$f\left(\lambda u + (1-\lambda)v\right) \leq \lambda f(u) + (1-\lambda) f(v).$$

Moreover, if this inequality holds $\forall\, u \neq v$ and $0 \leq \lambda \leq 1$, then f is said to be strictly convex.

Example 2.1.19 (Convex function)**.** *An example of a convex function is the total variation (TV) of a function $u\colon \Omega \subset \mathbb{R}^n \to \mathbb{R}$ (denoted by $TV(u)$), defined as followed*

$$TV(u) = \int_{\Omega} |\nabla u(\boldsymbol{x})|\, d\Omega.$$

2.2 Calculus of variations

In this section we consider a particular class of minimisation problem which is a very common occurrence in many real world imaging problems. The idea is to search for an appropriate function, rather than the value of a variable, which makes a given quantity (typically an energy integral in imaging problems) stationary. The method of calculus of variations aims to find the extrema of a given quantity (usually of the form of a definite integral), depending on some unknown function and possibly its derivatives.

2.2.1 Variation of a functional

Definition 2.2.1 (Admissible functions)**.** *An input function $\boldsymbol{u}(\boldsymbol{x})$, which is acceptable to a functional, is called admissible provided it satisfies the function smoothness condition and boundary conditions.*

Consider a general functional $E(\boldsymbol{u})\colon \mathcal{F} \to \mathbb{R}$ where $\mathcal{F}$ denotes some normed linear solution space consisting of admissible functions (for example $\mathcal{F} \subset \mathbb{R}^d$ where $d \geq 1$) with the following form

$$E(\boldsymbol{u}) = \int_{\boldsymbol{x}} \mathcal{L}\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}), \nabla \boldsymbol{u}(\boldsymbol{x})\big)\, d\boldsymbol{x}.$$

In other words the functional $E(\boldsymbol{u})$ depends upon the variable $\boldsymbol{x} = [x_1, \ldots, x_d]^T$, the unknown function $\boldsymbol{u}(\boldsymbol{x})$ and the gradient of the unknown function $\nabla \boldsymbol{u}(\boldsymbol{x})$. Then the method of calculus of variations aims to solve the following minimisation problem

$$\min_{\boldsymbol{u}} \big\{E(\boldsymbol{u})\big\}.$$

2.2.2 Gâteaux derivative of a functional

Definition 2.2.2 (Gâteaux derivative). *Let $\mathcal{F}$ define a Banach space, and also let $E(\boldsymbol{u})$ be a functional defined such that $E\colon \mathcal{F} \to \mathbb{R}$. Then the Gâteaux derivative of E is defined as*

$$\begin{aligned}\delta E(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\phi}(\boldsymbol{x})) &= \lim_{\varepsilon \to 0} \left\{ \frac{E(\boldsymbol{u}(\boldsymbol{x}) + \varepsilon \boldsymbol{\phi}(\boldsymbol{x})) - E(\boldsymbol{u}(\boldsymbol{x}))}{\varepsilon} \right\} \\ &= \frac{d}{d\varepsilon} E(\boldsymbol{u}(\boldsymbol{x}) + \varepsilon \boldsymbol{\phi}(\boldsymbol{x}))\Big|_{\varepsilon=0}.\end{aligned}$$

The limit denoted by $\delta E(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\phi}(\boldsymbol{x}))$ is also known as the first variation of E at $\boldsymbol{u}(\boldsymbol{x})$ in the direction of $\boldsymbol{\phi}(\boldsymbol{x})$ where

$$\boldsymbol{\phi}(\boldsymbol{x}) \in C_0^\infty(\Omega)$$

denotes a test function.

Definition 2.2.3 (Neighbourhood). *Let $\mathcal{F}$ define a solution space, $\boldsymbol{u}^* \in \mathcal{F}$ define a function and $0 < \varepsilon \in \mathbb{R}$ define a positive scalar. Then the neighbourhood of $\boldsymbol{u}^*$ (denoted by $N_\varepsilon(\boldsymbol{u}^*(\boldsymbol{x}))$), is defined as*

$$N_\varepsilon(\boldsymbol{u}^*(\boldsymbol{x})) = \{\boldsymbol{u} \in \Omega\colon |\boldsymbol{u} - \boldsymbol{u}^*| < \varepsilon\}.$$

Definition 2.2.4 (Local minimiser). *The functional $E\colon \mathcal{F} \to \mathbb{R}$ has a local minimiser $\boldsymbol{u}^*(\boldsymbol{x})$ if there exists a neighbourhood $N_\varepsilon(\boldsymbol{u}^*(\boldsymbol{x}))$ such that*

$$E(\boldsymbol{u}^*(\boldsymbol{x})) \leq E(\boldsymbol{u}(\boldsymbol{x})) \,\forall\, \boldsymbol{u}(\boldsymbol{x}) \in N_\varepsilon(\boldsymbol{u}^*(\boldsymbol{x})).$$

Definition 2.2.5 (Global minimiser). *The functional $E\colon \mathcal{F} \to \mathbb{R}$ has a global minimiser $\boldsymbol{u}^*(\boldsymbol{x})$ if*

$$E(\boldsymbol{u}^*(\boldsymbol{x})) \leq E(\boldsymbol{u}(\boldsymbol{x})) \,\forall\, \boldsymbol{u}(\boldsymbol{x}) \in \mathcal{F}.$$

Definition 2.2.6 (Stationary point). *Suppose at an admissible function $\boldsymbol{u}(\boldsymbol{x}) \in \mathcal{F}$ the functional E has zero first variation, in other words we have*

$$\delta E(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\phi}(\boldsymbol{x})) = 0, \,\forall\, \boldsymbol{\phi}(\boldsymbol{x}) \in C_0^\infty(\Omega).$$

Then the admissible function $\boldsymbol{u}(\boldsymbol{x})$ is said to be a stationary point of the functional E.

Theorem 2.2.7 (Necessary condition for a local minimiser). *For any minimiser of the functional $E\colon \mathcal{F} \to \mathbb{R}$, a necessary condition is the vanishing of the first variation δE. In other words we require*

$$\delta E(\boldsymbol{u}(\boldsymbol{x}); \boldsymbol{\phi}(\boldsymbol{x})) = \frac{d}{d\varepsilon} E(\boldsymbol{u}(\boldsymbol{x}) + \varepsilon \boldsymbol{\phi}(\boldsymbol{x}))\Big|_{\varepsilon=0} = 0.$$

2.2.3 The divergence theorem

The divergence theorem (or Gauss theorem) relates the flux of a vector field through a surface, to the divergence of the vector field inside the surface. Let us suppose $\Omega \subset \mathbb{R}^d$ is a compact subset of $\mathbb{R}^d$ which possesses a piecewise smooth boundary denoted by $\partial\Omega$, and also suppose

$$\boldsymbol{F} \equiv \boldsymbol{F}(\boldsymbol{x}) \in C^\infty(\Omega)$$

is a continuously differentiable vector field. Then

$$\int_\Omega \big(\nabla \cdot \boldsymbol{F}\big)\, d\Omega = \int_{\partial\Omega} \boldsymbol{F} \cdot \boldsymbol{n}\, dS \tag{2.1}$$

where $\nabla \cdot \boldsymbol{F}$ denotes the divergence of the vector field $\boldsymbol{F}$, and $\boldsymbol{n}$ denotes the outward unit normal of $\partial\Omega$.

2.2.4 Green's identities

One result of the divergence theorem is, depending on the form of the scalar field $\boldsymbol{F}$, we can derive several relations known as Green's identities.

Corollary 2.2.8 (Green's first identity)**.** *Given a vector field of the form*

$$\boldsymbol{F} = \mu\nabla\nu$$

where

$$\mu \in C^1(\Omega),\, \nu \in C^2(\Omega)$$

are scalar functions defined on $\Omega \subset \mathbb{R}^d$, then

$$\int_\Omega \big(\mu\Delta\nu + \nabla\mu \cdot \nabla\nu\big)\, d\Omega = \int_{\partial\Omega} \mu\big(\nabla\nu \cdot \boldsymbol{n}\big)\, dS \tag{2.2}$$

where Δ denotes the Laplace operator and $\boldsymbol{n}$ denotes the outward unit normal of the boundary $\partial\Omega$.

Corollary 2.2.9 (Green's second identity)**.** *Given a vector field of the form*

$$\boldsymbol{F} = \mu\varepsilon\nabla\nu - \nu\varepsilon\nabla\mu$$

where

$$\varepsilon \in C^1(\Omega) \text{ and } \mu,\, \nu \in C^2(\Omega)$$

are scalar functions. Then

$$\int_\Omega \Big[\mu\nabla \cdot \big(\varepsilon\nabla\nu\big) - \nu\nabla \cdot \big(\varepsilon\nabla\mu\big)\Big]\, d\Omega = \int_{\partial\Omega} \varepsilon\left(\mu\frac{\partial\nu}{\partial\boldsymbol{n}} - \nu\frac{\partial\mu}{\partial\boldsymbol{n}}\right) dS.$$

Or in the special case where $\varepsilon = 1$, we have

$$\int_\Omega \left(\mu\Delta\nu - \nu\Delta\mu\right) d\Omega = \int_{\partial\Omega} \left(\mu\frac{\partial\nu}{\partial\boldsymbol{n}} - \nu\frac{\partial\mu}{\partial\boldsymbol{n}}\right) dS \tag{2.3}$$

where $\frac{\partial\mu}{\partial\boldsymbol{n}}$ and $\frac{\partial\nu}{\partial\boldsymbol{n}}$ are equivalent to the directional derivatives of μ and ν in the direction of the outward unit normal $\boldsymbol{n}$ respectively, i.e.

$$\frac{\partial\mu}{\partial\boldsymbol{n}} \equiv \nabla\mu\cdot\boldsymbol{n},\ \frac{\partial\nu}{\partial\boldsymbol{n}} \equiv \nabla\nu\cdot\boldsymbol{n}.$$

Corollary 2.2.10 (Green's third identity)**.** *Given the special case of Corollary 2.2.9 where we have $\nu = G$, with G denoting the fundamental solution of the Laplace operator at the point $\boldsymbol{\eta}$ defined by*

$$\Delta G(\boldsymbol{x},\boldsymbol{\eta}) = \delta(\boldsymbol{x}-\boldsymbol{\eta}).$$

Then, if $\mu \in C^2(\Omega)$ is a scalar function defined on $\Omega \subset \mathbb{R}^d$, we have

$$\int_\Omega \left[G(\boldsymbol{x},\boldsymbol{\eta})\Delta\mu(\boldsymbol{x})\right] d\Omega - \mu(\boldsymbol{\eta}) = \int_{\partial\Omega}\left[G(\boldsymbol{x},\boldsymbol{\eta})\frac{\partial\mu}{\partial\boldsymbol{n}}(\boldsymbol{x}) - \mu(\boldsymbol{x})\frac{\partial G(\boldsymbol{x},\boldsymbol{\eta})}{\partial\boldsymbol{n}}\right] dS.$$

Corollary 2.2.11 (Integration by parts)**.** *Given a scalar function g and vector field $\boldsymbol{F}$, we can apply the divergence theorem* (2.1) *to the product $g\boldsymbol{F}$ which results in the following*

$$\int_\Omega \left(\boldsymbol{F}\cdot\nabla g + g\nabla\cdot\boldsymbol{F}\right) d\Omega = \int_{\partial\Omega} g\boldsymbol{F}\cdot\boldsymbol{n}\, dS. \tag{2.4}$$

In the 1D case we have

$$\boldsymbol{F} = u(x),\ g = v(x),\ \nabla\boldsymbol{F} = u'(x),\ \nabla g = v'(x).$$

Thus the integral (2.4) *reduces to the integration by parts formula given by*

$$\int_\Omega u(x)v'(x)\,dx = v(x)u(x) - \int_\Omega v(x)u'(x)\,dx.$$

2.2.5 Fundamental lemma of calculus of variations

From the method of calculus of variations, in order for a functional E to possess a minima, then it is necessary for its first variation δE to vanish. This necessary condition can be satisfied by first deriving, and then solving, the Euler-Lagrange (EL) equations. However, to do so we must use the fundamental lemma of calculus of variations (otherwise known as the Du Bois-Reymond lemma), which is given by the following:

Lemma 2.2.12 (Du Bois-Reymond lemma)**.** *Given some locally integrable and contin-*

uous function $\boldsymbol{u}$ *defined on the open interval* $\Omega \subset \mathbb{R}^d$*, then if*

$$\int_\Omega \boldsymbol{u}(\boldsymbol{x})\boldsymbol{\phi}(\boldsymbol{x})\, d\Omega = 0 \,\forall\, \boldsymbol{\phi}(\boldsymbol{x}) \in C_0^\infty(\Omega) \tag{2.5}$$

we have $\boldsymbol{u}(\boldsymbol{x}) = 0$*.*

Example 2.2.13. *Suppose we have the diffusion regularisation term given by*

$$E(u) = \frac{1}{2}\int_\Omega |\nabla u|^2 \, d\Omega \tag{2.6}$$

defined on the open interval $\Omega \subset \mathbb{R}^d$*. Also suppose we wish to determine the first variation of the functional* (2.6)*, then we begin by introducing the small positive parameter* ε *and test function* $\phi \in C_0^\infty(\Omega)$*. Next we introduce the small perturbation* $\varepsilon\phi$ *and compute the following*

$$\begin{aligned}
\delta E(u;\phi) &= \frac{1}{2}\frac{d}{d\varepsilon}\int_\Omega |\nabla(u+\varepsilon\phi)|^2 \, d\Omega\Big|_{\varepsilon=0} \\
&= \lim_{\varepsilon\to 0}\frac{1}{2\varepsilon}\int_\Omega |\nabla(u+\varepsilon\phi)|^2 - |\nabla u|^2 \, d\Omega \\
&= \lim_{\varepsilon\to 0}\frac{1}{2\varepsilon}\int_\Omega \big(u_{x_1}+\varepsilon\phi_{x_1}\big)^2 + \big(u_{x_2}+\varepsilon\phi_{x_2}\big)^2 - \big(u_{x_1}^2+u_{x_2}^2\big)\, d\Omega \\
&= \lim_{\varepsilon\to 0}\frac{1}{2\varepsilon}\int_\Omega u_{x_1}^2 + 2\varepsilon u_{x_1}\phi_{x_1} + u_{x_2}^2 + 2\varepsilon u_{x_2}\phi_{x_2} - \big(u_{x_1}^2+u_{x_2}^2\big) + \mathcal{O}(\varepsilon^2)\, d\Omega \\
&= \lim_{\varepsilon\to 0}\frac{1}{2\varepsilon}\int_\Omega 2\varepsilon\big(u_{x_1}\phi_{x_1} + u_{x_2}\phi_{x_2}\big) + \mathcal{O}(\varepsilon^2)\, d\Omega \\
&= \int_\Omega \nabla u\cdot\nabla\phi \, d\Omega.
\end{aligned} \tag{2.7}$$

Now in order for us to use the Du Bois-Reymond lemma (2.5)*, we must first get the integral* (2.7) *into the form*

$$\int_\Omega u\phi \, d\Omega.$$

We do this by applying Green's first identity (2.2)*, with the substitutions*

$$\phi = \mu,\ u = \nu.$$

Doing so gives

$$\int_\Omega \nabla u\cdot\nabla\phi \, d\Omega = \int_{\partial\Omega}\phi\big(\nabla u\cdot\boldsymbol{n}\big)\, dS - \int_\Omega \phi\Delta u \, d\Omega.$$

The EL equations for the functional E *are then derived by setting the first variation* δE *equal to zero, or in other words*

$$-\int_\Omega \phi\Delta u \, d\Omega + \int_{\partial\Omega}\phi\big(\nabla u\cdot\boldsymbol{n}\big)\, dS = 0. \tag{2.8}$$

After using the Du Bois-Reymond lemma (2.5) *on each integral separately in* (2.8)*, we*

obtain the following EL equations

$$\begin{cases} -\Delta u = 0 \text{ in } \Omega, \\ \nabla u \cdot \boldsymbol{n} = 0 \text{ on } \partial\Omega. \end{cases} \tag{2.9}$$

The solution of the partial differential equations (PDEs) (2.9) *yield a minimiser of the functional given in* (2.6).

2.2.6 Functions of bounded variation

Given a bounded open subset $\Omega \subset \mathbb{R}^d$ and $u \in L^1(\Omega)$, then the total variation (TV) of u is given by

$$TV(u) = \int_\Omega |Du| \, d\Omega = \sup_{\boldsymbol{\varphi} \in \boldsymbol{V}} \left\{ \int_\Omega u(\boldsymbol{x}) \nabla \cdot \boldsymbol{\varphi} \, d\boldsymbol{x} \right\}$$

where V denotes the set of test functions given by

$$V = \left\{ \boldsymbol{\varphi} = \left[\varphi_1, \ldots, \varphi_d\right]^T \in C_0^1\big(\Omega, \mathbb{R}^d\big) \colon \, \|\boldsymbol{\varphi}\|_{L^\infty} \leq 1 \, \forall \, i = 1, \ldots, d \right\}$$

with $\nabla \cdot \boldsymbol{\varphi}$ denoting the divergence defined by

$$\nabla \cdot \boldsymbol{\varphi} = \sum_{i=1,j=1}^{d} \frac{\partial \varphi_i}{\partial x_j}$$

and where $d\boldsymbol{x}$ denotes the Lebesgue measure.

Remark 2.2.14. *In Euclidean space, the Lebesgue measure is the standard way to assign a measure (e.g. length, area, volume) to a given subset. Hence, sets which have a finite Lebesgue measure are known as Lebesgue measurables. This is the measure which is used to define Lebesgue integration.*

There is an interesting case (as described in [57]) when $u \in C^1\big(\Omega, \mathbb{R}^d\big)$ and after the use of integration by parts we get

$$\int_\Omega u \nabla \cdot \boldsymbol{\varphi} \, d\boldsymbol{x} = -\int_\Omega \sum_{i=1}^{d} \frac{\partial u}{\partial x_i} \varphi_i \, d\boldsymbol{x}$$

for every

$$\boldsymbol{\varphi} \in C_0^1\big(\Omega, \mathbb{R}^d\big)$$

and

$$\int_\Omega |Du| \, d\boldsymbol{x} = \int_\Omega |\nabla u| \, d\Omega$$

with

$$\nabla u = \frac{\partial u}{\partial x_i}$$

for $i = 1, \ldots, d$. Then a function $u \in L^1(\Omega)$ is said to have bounded variation in Ω if the total variation of u is finite, i.e.

$$TV(u) < \infty.$$

The space of all such functions in $L^1(\Omega)$ is denoted by $BV(u)$.

Example 2.2.15. *Consider the following three 1D functions u_1, u_2 and u_3 defined by*

$$\begin{aligned} u_1(x_1) &= \cos(x_1) \\ u_2(x_1) &= x_1^2 \sin(x_1) \\ u_3(x_1) &= \begin{cases} 0 & x_1 = 0 \\ x_1 \sin\left(\frac{1}{x_1}\right) & x_1 \neq 0 \end{cases} \end{aligned}$$

with

$$x_1 \in \Omega = [0, 1].$$

Computing the TV of $u_1(x_1)$ and $u_2(x_1)$, we get

$$\begin{aligned} TV(u_1) &= \int_\Omega |\nabla u_1(x_1)| \, dx_1 \\ &= \int_0^1 \left| \frac{\partial u_1}{\partial x_1} \right| dx_1 = \int_0^1 |-\sin(x_1)| \, dx_1 = \underline{0.4597} \end{aligned}$$

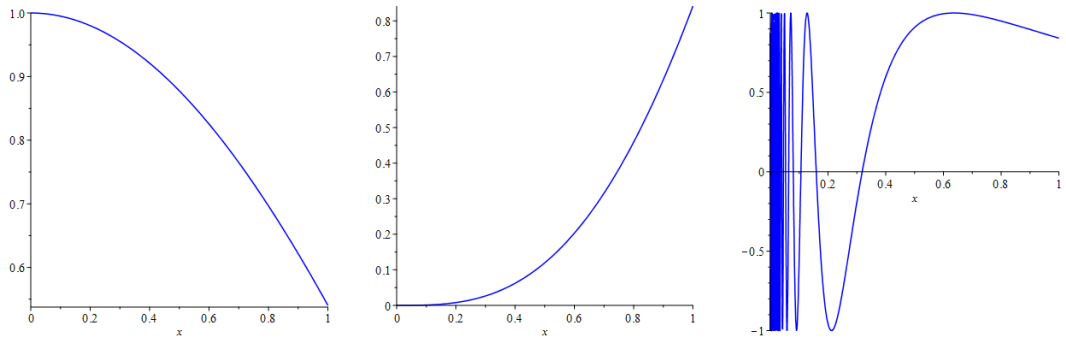
and

$$\begin{aligned} TV(u_2) &= \int_\Omega |\nabla u_2(x_1)| \, dx_1 \\ &= \int_0^1 \left| \frac{\partial u_2}{\partial x_1} \right| dx_1 = \int_0^1 \left| 2x_1 \sin(x_1) + x_1^2 \cos(x_1) \right| dx_1 = \underline{0.8415} \end{aligned}$$

respectively. Thus the functions $u_1(x_1)$ and $u_2(x_1)$ belong to the space of functions with bounded variations $BV(\Omega)$. The function $u_3(x_1)$ however, does not belong to the space $BV(\Omega)$ since it has infinite TV. This is shown in Figure 2.1(c) where we see the oscillations increase as $x_1 \to 0$, thus increasing the value of $TV\big(u_3\big)$.

2.3 Ill-posedness and regularisation

Ill-posed inverse problems are a common occurrence in many real-world applications such as astronomy [37, 59, 140], oceanography [128] and especially image processing [7, 50, 109]. In 1902, French mathematician Jaques Hadamard coined a definition to determine whether a problem is well-posed or not. According to Hadamard, for a



(a) $u_1(x_1)$. (b) $u_2(x_1)$. (c) $u_3(x_1)$.

Figure 2.1: From left to right we have the graphs of the functions $u_1(x_1)$, $u_2(x_1)$, $u_3(x_1)$ where the functions $u_1(x_1)$ and $u_2(x_1)$ are of bounded variation on the domain $\Omega = [0, 1]$. The function $u_3(x_1)$ however has infinite total variation, and is therefore is not of bounded variation on the domain $\Omega = [0, 1]$.

problem to be well-posed then the following three conditions must hold:

(i) **Existence.** The solution exists;

(ii) **Uniqueness.** The solutions is unique;

(iii) **Stability.** The solution depends continuously on the data, i.e. small changes in the data do not cause large changes in the solution.

If any of these conditions do not hold, then the problem is said to be ill-posed in the sense of Hadamard. In this section we consider ill-posed problems, and how such problems can be solved numerically through the use of Tikhonov regularisation [129].

2.3.1 Inverse problems

Definition 2.3.1 (Forward and inverse problems)**.** *Suppose we have the following*

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f} \tag{2.10}$$

with

$$\boldsymbol{A} \in L(\mathcal{H}, \mathcal{F}),\ \boldsymbol{u} \in \mathcal{H},\ \boldsymbol{f} \in \mathcal{F}$$

and where $\mathcal{H}, \mathcal{F}$ define Hilbert spaces. Then we define the forward problem to be the problem which determines the data $\boldsymbol{f}$ from the parameter $\boldsymbol{u}$ using the operator $\boldsymbol{A}$. In other words

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{u} \text{ for } \boldsymbol{u} \in \mathcal{H},\ \boldsymbol{f} \in \mathcal{F}. \tag{2.11}$$

An inverse problem, on the other hand, is defined to be the problem which determines the parameter $\boldsymbol{u} \in \mathcal{H}$ from the data $\boldsymbol{f} \in \mathcal{F}$ such that (2.11) *(or some approximation) holds.*

Definition 2.3.2 (Well-posedness)**.** *The problem* (2.10) *is said to be well-posed, in the sense of Hadamard, if the following conditions all hold:*

(i) **Existence.** $\forall\, \boldsymbol{f} \in \mathcal{F}$*, there exists a solution* $\boldsymbol{u} \in \mathcal{H}$ *such that* (2.10) *is true;*

(ii) **Uniqueness.** $\forall\, \boldsymbol{f} \in \mathcal{F}$*, the solution* $\boldsymbol{u} \in \mathcal{H}$ *is unique;*

(iii) **Stability.** *The solution* $\boldsymbol{u} \in \mathcal{H}$ *depends continuously on the data.*

If any of the above do not hold, then we say the problem (2.10) *is ill-posed in the sense of Hadamard.*

Example 2.3.3 (Ill-posed problem)**.** *Suppose we have the system shown in* (2.10) *with*

$$\boldsymbol{A} = \begin{bmatrix} 3 & 5 \\ 9 & 15 \end{bmatrix}, \boldsymbol{f} = \begin{bmatrix} 3 \\ 9 \end{bmatrix}, \boldsymbol{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{2.12}$$

We notice the two equations are not linearly independent, thus we say the system is under-determined (i.e. there are fewer equations than unknowns). Such under-determined systems are known to possess an infinite number of solutions, for example two possible solutions to the problem (2.12) *are*

$$[u_1, u_2]^T = \left[0, \frac{3}{5}\right]^T, \, [u_1, u_2]^T = [1, 0]^T.$$

Therefore the problem does not have a unique solution, and hence the problem is ill-posed in the sense of Hadamard.

2.3.2 Tikhonov regularisation

As we have mentioned, ill-posed problems are a very common occurrence in a lot of real-world image processing problems, one notable example being image registration. Therefore it is crucial we are able to solve such problems. In [129] Andrey Tikhonov proposed the idea of an additional constraint, known as a regularisation term, to be included in the original ill-posed problem in order to stabilise the solution. In general, the regularisation term is chosen according to some prior physical information on the solution.

Example 2.3.4 (Tikhonov regularisation)**.** *Let us consider the problem of registering a pair of images to one another according to the matching of intensity values. In practice this is typically done by minimising the sum of squared distances (SSD) between a given template image (which we look to deform) and a given reference image (which remains fixed) which we denote by*

$$T,\, R \in \Omega \subset \mathbb{R}^d$$

respectively. In other words, we are looking to solve the following

$$\min_{\boldsymbol{\varphi}} \left\{ \left| T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) - R(\boldsymbol{x}) \right|_2^2 \right\} \tag{2.13}$$

where

$$\boldsymbol{\varphi}(\boldsymbol{x}) \in \Omega \subset \mathbb{R}^d$$

denotes the transformation which matches the template image to the reference image, i.e.

$$T(\boldsymbol{\varphi}(\boldsymbol{x})) = R(\boldsymbol{x}).$$

Unfortunately, minimising the SSD measure alone is not sufficient to produce a unique solution, and thus (2.13) *is ill-posed. However, through the inclusion of an additional regularisation term we can ensure the solution obtained is unique. Hence the problem*

$$\min_{\boldsymbol{\varphi}} \left\{ \left| T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) - R(\boldsymbol{x}) \right|_2^2 + \alpha \left| \mathscr{R}\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) \right|_2^2 \right\}$$

where $\mathscr{R}$ denotes the regularisation operator and $\alpha \in \mathbb{R}^+$ is a weighting parameter between the two terms, is well-posed.

The general form of a Tikhonov regularisation model can be expressed in the following way

$$\min_{\boldsymbol{\varphi}} \left\{ \mathscr{D}(R, T, \boldsymbol{\varphi}) + \alpha \mathscr{R}(\boldsymbol{\varphi}) \right\}$$

where $\mathscr{D}$ denotes the similarity (or fitting) term and $\mathscr{R}$ the regularisation term, with $\alpha \in \mathbb{R}^+$ representing a weighting parameter between these two terms.

2.4 Discrete PDEs and notation

In general, when continuous partial differential equations (PDEs) are encountered, we must look to obtain a numerical approximation of the solution as typically analytical solutions are not possible to obtain. Suppose we have some open and bounded domain in $\mathbb{R}^d$ denoted by Ω with boundary given by $\Gamma = \partial\Omega$, then we can define a continuous linear d-dimensional boundary value problem (BVP) by the following

$$\begin{cases} \boldsymbol{\mathcal{L}}_\Omega \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}_\Omega(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Omega, \\ \boldsymbol{\mathcal{L}}_\Gamma \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}_\Gamma(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Gamma \end{cases} \tag{2.14}$$

where $\boldsymbol{\mathcal{L}}$ denotes some linear operator and

$$\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}) = [u_1(\boldsymbol{x}), \ldots, u_d(\boldsymbol{x})]^T$$

is the unknown function to be determined. Similarly we can define a non-linear BVP in the following way

$$\begin{cases} \boldsymbol{\mathcal{N}}_\Omega \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}_\Omega(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Omega, \\ \boldsymbol{\mathcal{N}}_\Gamma \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{F}_\Gamma(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Gamma \end{cases} \tag{2.15}$$

where $\boldsymbol{\mathcal{N}}$ denotes some non-linear operator.

Definition 2.4.1 (Laplace operator)**.** *The Laplace operator, or Laplacian, is a differential operator which is defined to be the divergence of the gradient of a function in Euclidean space and is typically denoted by Δ. Alternatively, the Laplace operator can also be denoted by ∇^2, $\nabla \cdot \nabla$ or $div(\nabla)$. Consider some scalar function $f(x_1, x_2)$, then we define the Laplacian of f by*

$$\Delta f(x_1, x_2) = \partial^2_{x_1} f + \partial^2_{x_2} f = \frac{\partial^2 f}{\partial x_1^2} + \frac{\partial^2 f}{\partial x_2^2}.$$

Example 2.4.2 (Poisson equation)**.** *Consider the Poisson equation in 2D given by the following*

$$-\Delta \boldsymbol{u}(\boldsymbol{x}) = F(\boldsymbol{x}) \text{ in } \Omega$$

with Neumann boundary conditions

$$\nabla \boldsymbol{u} \cdot \boldsymbol{n} = 0 \text{ on } \Gamma$$

where ∇ denotes the 2D spatial gradient operator and $\boldsymbol{n}$ the outward unit normal.

In image registration, the domain $\Omega \subset \mathbb{R}^2$ is typically rectangular with known values f at points uniformly distributed throughout the domain. For this reason discretisation based upon a finite difference method (FDM) is typically favoured over other discretisation methods such as the finite element method (FEM) [3, 77].

In the 2D setting let us assume

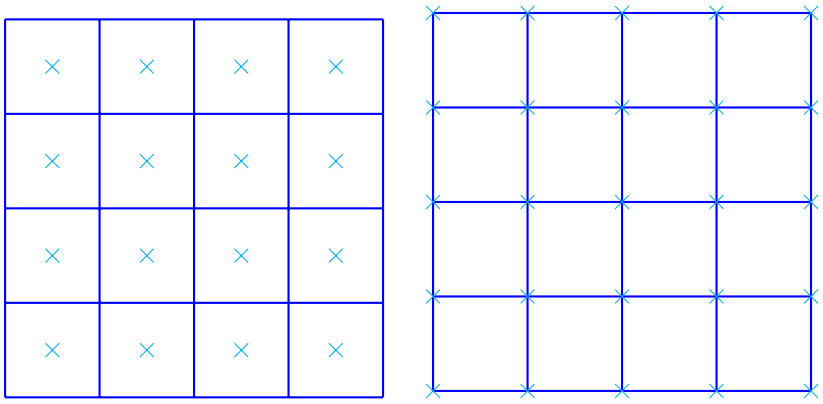
$$\Omega = (a, b) \times (c, d)$$

is rectangular, then we impose a Cartesian mesh (or grid) with the following spacing

$$h_1 = \frac{b-a}{n_1}, \; h_2 = \frac{d-c}{n_2}$$

in the x_1 and x_2 directions respectively. Now there are two ways in which we can discretise the mesh, these are vertex-centred discretisation and cell-centred discretisation. In vertex-centred discretisation, the grid points are placed at the vertices of the mesh such that there are $(n_1 + 1) \times (n_2 + 1)$ points including those on the boundary. For this case, the grid point (i, j) is located at

$$(x_{1_i}, x_{2_j}) = (ih_1, jh_2) \text{ for } 0 \leq i \leq n_1, \; 0 \leq j \leq n_2.$$



(a) Cell-centred grid. (b) Vertex-centred grid.

Figure 2.2: Visual representation of a cell-centred and vertex-centred discretisation for a square domain.

In cell-centred discretisation, the grid points are placed at the centre of cells such that there are $n_1 \times n_2$ points. For this case, the grid point (i, j) is located at

$$(x_{1_i}, x_{2_j}) = \left(a + \frac{2i-1}{2}h_1, c + \frac{2j-1}{2}h_2\right) \text{ for } 0 \leq i \leq n_1,\ 0 \leq j \leq n_2.$$

In either case the interior of the mesh is denoted by Ω^h and the boundary by either Γ^h or $\partial\Omega^h$. Figure 2.2 shows an example of a vertex-centred and cell-centred discretisation applied to a square mesh. After the discretisation method has been decided, any operators in the PDE can be approximated locally using Taylor expansions given by the following

$$u(x_1 + h_1, x_2) = u(x_1, x_2) + h_1\frac{\partial u}{\partial x_1}(x_1, x_2) + \frac{h_1^2}{2}\frac{\partial^2 u}{\partial x_1^2}(x_1, x_2) + \mathcal{O}(h_1^3)$$

and

$$u(x_1 - h_1, x_2) = u(x_1, x_2) + h_1\frac{\partial u}{\partial x_1}(x_1, x_2) - \frac{h_1^2}{2}\frac{\partial^2 u}{\partial x_1^2}(x_1, x_2) + \mathcal{O}(h_1^3)$$

where $\mathcal{O}(h_1^3)$ denotes the terms containing powers of h_1 of order 3 and above. Now the operator $\frac{\partial u}{\partial x_1}(x_1, x_2)$ can also be approximated locally in three different ways. These are the first order forward difference, first order backward difference and first order central difference and are defined by the following:

First order forward difference.

$$\delta^+_{x_1}(u)_{i,j} = \frac{(u)_{i+1,j} - (u)_{i,j}}{h_1} \approx \left(\frac{\partial u}{\partial x_1}\right)_{i,j};$$

First order backward difference.

$$\delta^{-}_{x_1}(u)_{i,j} = \frac{(u)_{i,j} - (u)_{i-1,j}}{h_1} \approx \left(\frac{\partial u}{\partial x_1}\right)_{i,j};$$

First order central difference.

$$\delta^{c}_{x_1}(u)_{i,j} = \frac{(u)_{i+1,j} - (u)_{i-1,j}}{2h_1} \approx \left(\frac{\partial u}{\partial x_1}\right)_{i,j}.$$

Similarly higher order derivatives can be approximated in a similar manner, for example the second order central difference approximation of the derivative $\frac{\partial^2 u}{\partial x_1^2}(x_1, x_2)$ is given by

$$\delta^{c}_{x_1 x_1}(u)_{i,j} = \frac{(u)_{i-1,j} - 2(u)_{i,j} + (u)_{i+1,j}}{h_1^2} \approx \left(\frac{\partial^2 u}{\partial x_1^2}\right)_{i,j}. \tag{2.16}$$

Moreover, we denote the discrete versions of (2.14) and (2.15) by

$$\boldsymbol{\mathcal{L}}^h \boldsymbol{u}^h = \boldsymbol{F}^h, \boldsymbol{\mathcal{N}}^h \boldsymbol{u}^h = \boldsymbol{F}^h$$

respectively.

Remark 2.4.3. *In image registration, the domain Ω^h represents an image domain. While both vertex-centred discretisations (e.g. [12–16, 18]) and cell-centred discretisations (e.g. [32–35, 67]) are used, the choice of which discretisation to use is typically a matter of preference. Moreover, while the choice of the image domain Ω^h can be arbitrary, there are commonly only two possibilities which are used in practice. The first choice is to define Ω^h such that the grid spacing is equal to 1 in each dimension, for example if we have an image of size 256×128 then we would take*

$$\Omega^h = [0, 256] \times [0, 128].$$

The second choice is to define Ω^h as the unit square, in other words we take

$$\Omega^h = [0, 1] \times [0, 1]$$

with grid spacing

$$h_1 = \frac{1}{n_1 - 1}, h_2 = \frac{1}{n_2 - 1}$$

in the x_1, x_2 directions respectively. The latter is the most common choice in the literature, and is indeed the choice we use throughout this thesis.

2.4.1 Stencil notation

Consider the discrete Poisson equation defined on the unit square $\Omega^h = [0,1]^2$ with interval h and Dirichlet boundary conditions, in other words

$$\begin{cases} -\left(\Delta^h \boldsymbol{u}^h\right)_{i,j} = \left(\boldsymbol{F}^h\right)_{i,j} \text{ in } \Omega^h, \\ \left(\boldsymbol{u}^h\right)_{i,j} = \left(\boldsymbol{F}^h\right)_{i,j} = 0 \text{ on } \Gamma^h. \end{cases} \tag{2.17}$$

Now the discrete Laplace operator $\left(\Delta^h \boldsymbol{u}^h\right)_{i,j}$ can be written, using the second order difference approximation (2.16), in the following way

$$\begin{aligned} \left(\Delta^h \boldsymbol{u}^h\right)_{i,j} &= \delta^{ch}_{x_1 x_1}\left(\boldsymbol{u}^h\right)_{i,j} + \delta^{ch}_{x_2 x_2}\left(\boldsymbol{u}^h\right)_{i,j} \\ &= \frac{1}{h^2}\left[\left(\boldsymbol{u}^h\right)_{i-1,j} - 2\left(\boldsymbol{u}^h\right)_{i,j} + \left(\boldsymbol{u}^h\right)_{i+1,j}\right] \\ &+ \frac{1}{h^2}\left[\left(\boldsymbol{u}^h\right)_{i,j-1} - 2\left(\boldsymbol{u}^h\right)_{i,j} + \left(\boldsymbol{u}^h\right)_{i,j+1}\right] \\ &= \frac{1}{h^2}\left[\left(\boldsymbol{u}^h\right)_{i,j-1} + \left(\boldsymbol{u}^h\right)_{i-1,j} - 4\left(\boldsymbol{u}^h\right)_{i,j} + \left(\boldsymbol{u}^h\right)_{i+1,j} + \left(\boldsymbol{u}^h\right)_{i,j+1}\right] \end{aligned} \tag{2.18}$$

which can be written using the following stencil notation

$$\left(\Delta^h \boldsymbol{u}^h\right)_{i,j} = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}\left(\boldsymbol{u}^h\right)_{i,j}. \tag{2.19}$$

2.4.2 Matrix notation

An alternate way of writing the discrete system

$$\boldsymbol{\mathcal{L}}^h \boldsymbol{u}^h = \boldsymbol{F}^h$$

is to use matrix notation. We do this by stacking the grid functions $\boldsymbol{u}^h$ along rows, starting from the bottom left (labelled 1 in Figure 2.3) and ending with the top right point (labelled $(n-2)^2$ in Figure 2.3), to form a column vector $\bar{\boldsymbol{u}}^h$. Such a method of ordering is known as lexicographical ordering, and an example can be seen in Figure 2.3. The vector on the right hand side is stacked in much the same way to produce a column vector $\bar{\boldsymbol{F}}^h$. Then the discrete linear system can be written in the form

$$\boldsymbol{A}^h \bar{\boldsymbol{u}}^h = \bar{\boldsymbol{F}}^h.$$

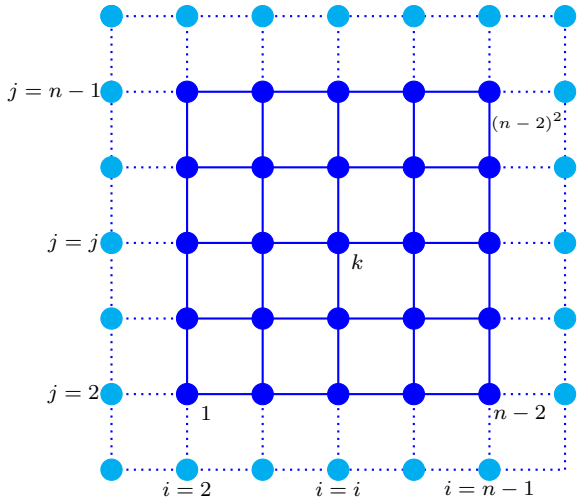


Figure 2.3: Illustration of a lexicographic ordering system. The indexing on the interior points show how the discrete interior points are ordered in a lexicographic system. The solid dark blue lines represent the interior Ω^h with points indicated by the solid dark blue circles, while the dashed light blue lines denote the boundary $\partial\Omega^h$ with points indicated by the solid light blue circles.

Example 2.4.4. *Returning to the Poisson equation in* (2.17)*, we can see from* (2.18) *for a general row* k *of the system matrix* $\boldsymbol{A}^h$ *the diagonal elements are*

$$a_{k,k} = \frac{4}{h^2}$$

and the off-diagonal elements are

$$a_{k\pm1,k} = a_{k,k\pm1} = -\frac{1}{h^2}$$

with all remaining entries being zero, and where appropriate modifications have been made for boundary points. This results in $\boldsymbol{A}^h$ *being a* $(n-2)^2 \times (n-2)^2$ *block tri-diagonal matrix consisting of blocks of size* $(n-2) \times (n-2)$*.*

2.4.3 Boundary conditions

Commonly there are two types of boundary conditions (BCs) which arise in PDEs, these are Dirichlet BCs and Neumann BCs. Dirichlet BCs specify the values of the function which must be satisfied on the boundary, while Neumann BCs specify values which the normal derivative of the function on a surface must satisfy. Again looking back to the Poisson equation (2.17), if we have Neumann BCs

$$\nabla \boldsymbol{u} \cdot \boldsymbol{n} = 0$$

instead of Dirichlet BCs, we need to access so-called 'ghost' points when we are sufficiently close to the boundary. For example, if we consider points on the right boundary

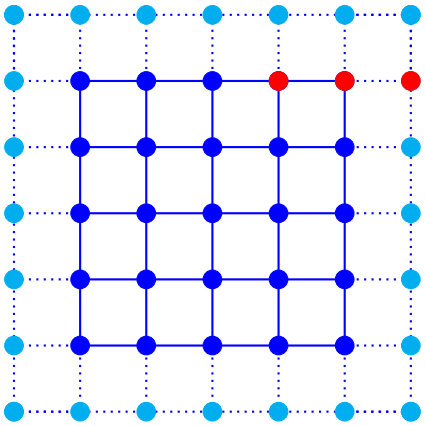


Figure 2.4: Illustration of how ghost points outside of the domain Ω^h, using a vertex-centred discretisation, are defined. The solid blue lines represent the discretised domain Ω^h with solid blue circles representing the vertex-centred grid points, while the dashed light blue lines and solid light blue circles represent the ghost points outside of the domain Ω^h. The solid red circles indicate the points used in the FD of the example point show in (2.20)

we have

$$\frac{\left(\boldsymbol{u}^h\right)_{n+1,j} - \left(\boldsymbol{u}^h\right)_{n-1,j}}{2h} = 0 \tag{2.20}$$

and we see the 'ghost' point $\left(\boldsymbol{u}^h\right)_{n+1,j}$ must be accessed, which is outside of the mesh. A visual representation of 'ghost' points can be seen in Figure 2.4. Along the right boundary, using stencil notation, we can write

$$-\left(\Delta^h \boldsymbol{u}^h\right)_{i,j} = \frac{1}{h^2}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 3 & 0 \\ 0 & -1 & 0 \end{bmatrix}\left(\boldsymbol{u}^h\right)_{i,j} = \left(\boldsymbol{F}^h\right)_{i,j}$$

where we have assumed Ω^h is given by the unit square.

2.4.4 Non-linear equations

For non-linear PDEs we can use the same treatment which we used for linear PDEs, namely we approximate the non-linear equation locally on a discrete mesh using a FDM. We denote the discrete form of the non-linear equation by the following

$$\begin{cases} \boldsymbol{\mathcal{N}}_{\Omega}^h \boldsymbol{u}^h(\boldsymbol{x}) = \boldsymbol{F}_{\Omega}^h(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Omega^h, \\ \boldsymbol{\mathcal{N}}_{\Gamma}^h \boldsymbol{u}^h(\boldsymbol{x}) = \boldsymbol{F}_{\Gamma}^h(\boldsymbol{x}) \text{ for } \boldsymbol{x} = [x_1, \ldots, x_d]^T \in \Gamma^h. \end{cases}$$

It is possible to write the non-linear equations using the matrix notation shown in §2.4.2, in this case the non-linear equations would take on the form

$$\boldsymbol{A}^h\big[\boldsymbol{u}^h\big]\bar{\boldsymbol{u}}^h = \bar{\boldsymbol{F}}^h. \tag{2.21}$$

2.5 Iterative methods

In this section we give a brief overview of some iterative methods which can be used to solve linear systems of the form shown in (2.21), where

$$\boldsymbol{A} \in \mathbb{R}^{n\times n},\ \boldsymbol{x},\ \boldsymbol{b} \in \mathbb{R}^{n\times 1}.$$

For the vast majority of applications where linear systems are required to be solved, the system is far too large to solve via direct methods, and so iterative methods are required to approximate the solution.

Suppose we are given a general system of linear equations with the form shown in (2.21). Then iterative methods, starting with some initial guess $\boldsymbol{x}^{(0)}$, aim to generate a sequence $\left\{\boldsymbol{x}^{(l)}\right\}_{l=1}^{\infty}$ from the recurrence relation

$$\boldsymbol{x}^{(l)} = \boldsymbol{T}\boldsymbol{x}^{(l-1)} + \boldsymbol{c} \tag{2.22}$$

where the matrix $\boldsymbol{T}$ and vector $\boldsymbol{c}$ are given by

$$\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{N},\ \boldsymbol{c} = \boldsymbol{M}^{-1}\boldsymbol{b} \tag{2.23}$$

and the matrices $\boldsymbol{M}$, $\boldsymbol{N}$ are obtained by splitting the matrix $\boldsymbol{A}$ according to

$$\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$$

where $\boldsymbol{M}$ is non-singular. For every step which computes $\boldsymbol{x}^{(l)}$ from $\boldsymbol{x}^{(l-1)}$, we typically refer to this as an iteration step or relaxation sweep.

2.5.1 Jacobi method

The Jacobi method solves the i^{th} equation of the linear system (2.21) using the following

$$x_i = \sum_{j=1, i\neq j}^{n}\left(-\frac{a_{ij}x_i}{a_{ii}}\right) + \frac{b_i}{a_{ii}} \text{ for } i = 1,\ldots,n.$$

Moreover, given $\boldsymbol{x}^{(l-1)}$ $(l \geq 1)$ the update $\boldsymbol{x}^{(l)}$ is computed using

$$x_i^{(l)} = \sum_{j=1, i\neq j}^{n}\left(-\frac{a_{ij}x_i^{(l-1)}}{a_{ii}}\right) + \frac{b_i}{a_{ii}} \text{ for } i = 1,\ldots,n$$

Algorithm 1 $[\boldsymbol{x}] \leftarrow Jacobi(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}^{(0)}, IMAX, Tol)$

1: Set $l = 1$, $n = size(\boldsymbol{x}^{(0)}, 1)$
2: **for** $l = 1, \ldots, IMAX$ **do**
3: **for** $i = 1, \ldots, n$ **do**
4: Set

$$x_i^{(l)} = \sum_{j=1, i \neq j}^{n} \left(-\frac{a_{ij} x_i^{(l-1)}}{a_{ii}} \right) + \frac{b_i}{a_{ii}} \tag{2.24}$$

5: **end for**
6: **if** $\left| \boldsymbol{A}\boldsymbol{x}^{(l)} - \boldsymbol{b} \right|_2 < Tol$ or $\left| \boldsymbol{x}^{(l)} - \boldsymbol{x}^{(l-1)} \right|_2 < Tol$ **then** Exit **else** Continue
7: **end for**

where we have assumed

$$a_{ii} \neq 0.$$

However, supposing one or more a_{ii} are zero, then we can use a re-ordering such that

$$a_{ii} \neq 0 \, \forall \, i = 1, \ldots, n.$$

We do this by splitting the matrix $\boldsymbol{A}$ into a diagonal part $\boldsymbol{D}$, strictly lower-triangular part $\boldsymbol{L}$ and strictly upper-triangular part $\boldsymbol{U}$ via the relation

$$\boldsymbol{A} = \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}.$$

Then the linear system (2.21) can be written as

$$\big[\boldsymbol{D} - \boldsymbol{L} - \boldsymbol{U}\big]\boldsymbol{x} = \boldsymbol{b} \iff \boldsymbol{x} = \boldsymbol{D}^{-1}\big[\boldsymbol{L} + \boldsymbol{U}\big]\boldsymbol{x} + \boldsymbol{D}^{-1}\boldsymbol{b}$$

which we see is of the same form as (2.23) with

$$\boldsymbol{T} = \boldsymbol{D}^{-1}\big[\boldsymbol{L} + \boldsymbol{U}\big], \, \boldsymbol{c} = \boldsymbol{D}^{-1}\boldsymbol{b}.$$

Then the matrix form of the Jacobi method can be written in the following way

$$\boldsymbol{x}^{(l)} = \boldsymbol{T}_J \boldsymbol{x}^{(l-1)} + \boldsymbol{c}_J \tag{2.25}$$

where

$$\boldsymbol{T}_J = \boldsymbol{D}^{-1}\big[\boldsymbol{L} + \boldsymbol{U}\big], \, \boldsymbol{c}_J = \boldsymbol{D}^{-1}\boldsymbol{b}.$$

The algorithm for solving (2.21), using the Jacobi method, is shown in Algorithm 1

An alternative to the Jacobi method, is the so-called weighted Jacobi method. In this method an intermediate value $\bar{\boldsymbol{x}}^{(l)}$ is computed using the current approximation $\boldsymbol{x}^{(l-1)}$

via the following relation

$$\bar{x}_i^{(l)} = \sum_{j=1, i\neq j}^{n} \left(-\frac{a_{ij} x_j^{(l-1)}}{a_{ii}} \right) + \frac{b_i}{a_{ii}} \text{ for } i = 1, \ldots, n \tag{2.26}$$

and then the new approximation $\boldsymbol{x}^{(l)}$ is computed according to

$$\boldsymbol{x}^{(l)} = \big(1-\omega\big)\boldsymbol{x}^{(l-1)} + \omega \bar{\boldsymbol{x}}^{(l)} \tag{2.27}$$

where

$$0 < \omega \in \mathbb{R}$$

is a weighting parameter to be selected. Here we remark if $\omega = 1$ in (2.27) then the weighted Jacobi method reduces to the Jacobi method. Using matrix form, the weighted Jacobi method can be written as

$$\begin{aligned} \boldsymbol{x}^{(l)} &= \left[\big(1-\omega\big)\boldsymbol{I} + \omega \boldsymbol{T}_J\right] \boldsymbol{x}^{(l-1)} + \omega \boldsymbol{c}_J \\ &\equiv \boldsymbol{T}_\omega \boldsymbol{x}^{(l-1)} + \boldsymbol{c}_\omega \end{aligned}$$

where $\boldsymbol{I}$ denotes the identity matrix and $\boldsymbol{T}_J$, $\boldsymbol{c}_J$ are as given in (2.25).

2.5.2 Gauss-Seidel method

The Gauss-Seidel (GS) method is an improvement to the Jacobi method, whereby the most recent updates are used to compute the value $x_i^{(l)}$ (rather than only using the old approximations). In other words we use the values

$$x_1^{(l)}, \ldots, x_{i-1}^{(l)}$$

instead of the values

$$x_1^{(l-1)}, \ldots, x_{i-1}^{(l-1)}$$

to compute the update $x_i^{(l)}$. Then the update $x_i^{(l)}$ is computed according to

$$x_i^{(l)} = -\frac{1}{a_{ii}} \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(l)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(l-1)} \right) + \frac{b_i}{a_{ii}} \text{ for } i = 1, \ldots, n \tag{2.28}$$

which can be re-written in the following way

$$a_{ii} x_i^{(l)} + \sum_{j=1}^{i-1} a_{ij} x_j^{(l)} = -\sum_{j=i+1}^{n} a_{ij} x_j^{(l-1)} + b_i. \tag{2.29}$$

Therefore we can write the GS method in the following matrix form

$$\begin{aligned}\big[\boldsymbol{D}-\boldsymbol{L}\big]\boldsymbol{x}^{(l)} &= \boldsymbol{U}\boldsymbol{x}^{(l-1)}+\boldsymbol{b}\\ \iff \boldsymbol{x}^{(l)} &= \big[\boldsymbol{D}-\boldsymbol{L}\big]^{-1}\boldsymbol{U}\boldsymbol{x}^{(l-1)}+\big[\boldsymbol{D}-\boldsymbol{L}\big]^{-1}\boldsymbol{b}\\ &\equiv \boldsymbol{T}_{GS}\boldsymbol{x}^{(l-1)}+\boldsymbol{c}_{GS}.\end{aligned} \tag{2.30}$$

The algorithm for the GS method is near identical to the one corresponding to the Jacobi method (i.e. Algorithm 1), with the exception equation (2.24) in Algorithm 1 is replaced with equation (2.28).

2.5.3 SOR method

The successive over relaxation (SOR) method is similar to the weighted Jacobi method, except applied to the GS method. Given the current approximation $x_i^{(l-1)}$, the intermediate values $\bar{x}_i^{(l)}$ are determined using the GS method, and then the values $\boldsymbol{x}^{(l)}$ are computed via

$$\boldsymbol{x}^{(l)} = \big(1-\omega\big)\boldsymbol{x}^{(l-1)}+\omega\bar{\boldsymbol{x}}^{(l)}$$

where

$$0<\omega<2$$

is some weighting parameter. If $0<\omega<1$ the scheme is referred to as under-relaxation and is used if the GS method does not converge, while if $1<\omega<2$ the scheme is referred to as over-relaxation and is used to accelerate the convergence of systems for which the GS method also converges. Also note if $\omega=1$, then the SOR method reduces simply to the GS method. The SOR method can also be written in matrix form using the following matrix splitting

$$\omega\boldsymbol{A} = \big[\boldsymbol{D}-\omega\boldsymbol{L}\big]-\big[\omega\boldsymbol{U}+\big(1-\omega\big)\boldsymbol{D}\big]$$

and can be expressed using the following relation

$$\begin{aligned}\boldsymbol{x}^{(l)} &= \big[\big(\boldsymbol{D}-\omega\boldsymbol{L}\big)-\big(\omega\boldsymbol{U}+\big(1-\omega\big)\boldsymbol{D}\big)\big]\,\boldsymbol{x}^{(l-1)}+\omega\big[\boldsymbol{D}-\omega\boldsymbol{L}\big]^{-1}\boldsymbol{b}\\ &\equiv \boldsymbol{T}_{SOR}\boldsymbol{x}^{(l-1)}+\boldsymbol{c}_{SOR}.\end{aligned}$$

2.5.4 Block methods

Suppose the vectors $\boldsymbol{x}$ and $\boldsymbol{b}$ are partitioned into several sub-vectors, in other words we have

$$\boldsymbol{x}=\big[\boldsymbol{x}_1^T,\ldots,\boldsymbol{x}_n^T\big],\ \boldsymbol{b}=\big[\boldsymbol{b}_1^T,\ldots,\boldsymbol{b}_n^T\big].$$

Then we can write the system (2.21) in the following block form

$$\begin{bmatrix} \boldsymbol{A}_{11} & \dots & \boldsymbol{A}_{1n} \\ \vdots & & \vdots \\ \boldsymbol{A}_{n1} & \dots & \boldsymbol{A}_{nn} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_n \end{bmatrix} \tag{2.31}$$

where the blocks $\boldsymbol{A}_{ij}$ are of size $n_p \times n_q$, and the blocks $\boldsymbol{x}_j$, $\boldsymbol{b}_j$ are of size $n_q \times 1$. Under the assumption the diagonal blocks $\boldsymbol{A}_{ii}$ are non-singular, we can extend the Jacobi and GS methods to be used on the block system (2.31). For the block Jacobi method, we update the block $\boldsymbol{x}_i$ according to the following relation

$$\boldsymbol{x}_i^{(l)} = \boldsymbol{A}_{ii}^{-1} \left[\sum_{j=1, i \neq j}^{n} -\boldsymbol{A}_{ij} \boldsymbol{x}_j^{(l-1)} + \boldsymbol{b}_i \right]$$

for $i = 1, \dots, n$. Similarly for the block GS method, we update the block $\boldsymbol{x}_i$ via the following

$$\boldsymbol{x}_i^{(l)} = \boldsymbol{A}_{ii}^{-1} \left[\sum_{j=1}^{i-1} -\boldsymbol{A}_{ij} \boldsymbol{x}_j^{(l)} + \sum_{j=i+1}^{n} -\boldsymbol{A}_{ij} \boldsymbol{x}^{(l-1)} + \boldsymbol{b}_i \right].$$

Now we remark since we have to invert the matrix $\boldsymbol{A}_{ii}$ in order to update each block $\boldsymbol{x}_i^{(l)}$, then naturally the larger the block $\boldsymbol{x}_i$ is the more expensive the update is to compute. However, while the block methods are more expensive computationally than their point relaxation counterparts (per relaxation sweep), they may benefit from an improved rate of convergence and therefore require fewer iterations to converge. Similar to the point relaxation schemes, we can also express the block variations using matrix notation. For the block Jacobi method we have

$$\begin{aligned} \boldsymbol{x}^{(l)} &= \boldsymbol{D}_B^{-1} \big[\boldsymbol{U}_B + \boldsymbol{L}_B \big] \boldsymbol{x}^{(l-1)} + \boldsymbol{D}_B^{-1} \boldsymbol{b} \\ &\equiv \boldsymbol{T}_{BJ} \boldsymbol{x}^{(l-1)} + \boldsymbol{c}_{BJ} \end{aligned}$$

and for the block GS method we have

$$\begin{aligned} \boldsymbol{x}^{(l)} &= \big[\boldsymbol{D}_B - \boldsymbol{L}_B \big] \boldsymbol{U}_B \boldsymbol{x}^{(l-1)} + \big[\boldsymbol{D}_B - \boldsymbol{L}_B \big] \boldsymbol{b} \\ &\equiv \boldsymbol{T}_{BGS} \boldsymbol{x}^{(l-1)} + \boldsymbol{c}_{BGS}. \end{aligned}$$

Example 2.5.1 (Line Gauss-Seidel)**.** *An example of a block method is the line GS method, where each block $\boldsymbol{x}_i$ in* (2.31) *corresponds to an entire row of $\boldsymbol{x}$ values from the discrete mesh which we update simultaneously.*

2.5.5 Convergence

In this section we show the sequence $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ converges to the true solution (of the linear system (2.21)) given by $\boldsymbol{x}$, where

$$\boldsymbol{x}^{(l)} = \boldsymbol{T}\boldsymbol{x}^{(l-1)} + \boldsymbol{c}$$

if and only if the spectral radius of the matrix $\boldsymbol{T}$ is less than 1 i.e.

$$\rho\big(\boldsymbol{T}\big) < 1.$$

Definition 2.5.2 (Spectral radius)**.** *Given some matrix $\boldsymbol{M}$, then we define the spectral radius of $\boldsymbol{M}$ by the following*

$$\rho\big(\boldsymbol{M}\big) = \max|\lambda|$$

where λ denotes the eigenvalues of $\boldsymbol{M}$.

Definition 2.5.3 (Convergent matrix)**.** *A square matrix $\boldsymbol{M}$ is said to be convergent if*

$$\lim_{k\to\infty}\Big\{\boldsymbol{M}^k\Big\} = 0.$$

Theorem 2.5.4 (Convergence of a matrix)**.** *A matrix $\boldsymbol{M}$ is a convergent matrix if and only if the spectral radius of $\boldsymbol{M}$ is less than 1, in other words*

$$\rho\big(\boldsymbol{M}\big) < 1. \tag{2.32}$$

Proof. A proof of Theorem 2.5.4 can be found in [118]. □

Lemma 2.5.5. *Suppose the spectral radius satisfies* (2.32)*, then the inverse*

$$\big[\boldsymbol{I} - \boldsymbol{M}\big]^{-1} = \sum_{k=0}^{\infty}\boldsymbol{M}^k$$

exists where $\boldsymbol{I}$ denotes the identity matrix.

Proof. Suppose λ denotes an eigenvalue of the matrix $\boldsymbol{M}$, then this implies $\big(1-\lambda\big)$ is an eigenvalue of the inverse matrix $\big[\boldsymbol{I} - \boldsymbol{M}\big]^{-1}$. Now using the fact (2.32) is true (from Theorem 2.5.4), implies $\lambda = 1$ is not an eigenvalue of $\boldsymbol{M}$, then we know

$$\big[1 - \lambda\big] = 0$$

is not an eigenvalue of $\big[\boldsymbol{I} - \boldsymbol{M}\big]^{-1}$ and hence $\big[\boldsymbol{I} - \boldsymbol{M}\big]^{-1}$ is not singular. Next let us

define the sequence $\mathcal{S}_m$ by the following

$$\begin{aligned} \mathcal{S}_m &= \boldsymbol{I} + \boldsymbol{M} + \cdots + \boldsymbol{M}^m \\ \implies \big[\boldsymbol{I} - \boldsymbol{M}\big]\mathcal{S}_m &= \big[\boldsymbol{I} + \boldsymbol{M} + \cdots + \boldsymbol{M}^m\big] - \big[\boldsymbol{M} + \cdots + \boldsymbol{M}^{m+1}\big] \\ &= \boldsymbol{I} - \boldsymbol{M}^{m+1}. \end{aligned}$$

From Theorem 2.5.4 we know (2.32) implies $\boldsymbol{M}$ is a convergent matrix, and therefore

$$\begin{aligned} \lim_{m\to\infty}\big\{\big[\boldsymbol{I} - \boldsymbol{M}\big]\mathcal{S}_m\big\} &= \lim_{m\to\infty}\big\{\big[\boldsymbol{I} - \boldsymbol{M}^{m+1}\big]\big\} = \boldsymbol{I} \\ \implies \big[\boldsymbol{I} - \boldsymbol{M}\big]^{-1} &= \lim_{m\to\infty}\{\mathcal{S}_m\} = \sum_{k=0}^{\infty}\boldsymbol{M}^k. \end{aligned}$$

□

Theorem 2.5.6 (Convergence of a sequence)**.** *Given some*

$$\boldsymbol{x}^{(0)} \in \mathbb{R}^n$$

then the sequence $\big\{\boldsymbol{x}^{(l)}\big\}_{l=0}^{\infty}$ *which is defined by*

$$\boldsymbol{x}^{(l)} = \boldsymbol{T}\boldsymbol{x}^{(l-1)} + \boldsymbol{c},\ l > 0$$

converges to the unique solution $\boldsymbol{x}$ *of*

$$\boldsymbol{x} = \boldsymbol{T}\boldsymbol{x} + \boldsymbol{c} \tag{2.33}$$

if and only if the spectral radius of $\boldsymbol{T}$ *is less than 1 i.e.*

$$\rho\big(\boldsymbol{T}\big) < 1. \tag{2.34}$$

Proof. Using the assumption (2.34), then we have

$$\begin{aligned} \boldsymbol{x}^{(l)} &= \boldsymbol{T}\boldsymbol{x}^{(l-1)} + \boldsymbol{c} \\ &= \boldsymbol{T}\big[\boldsymbol{T}\boldsymbol{x}^{(l-2)} + \boldsymbol{c}\big] + \boldsymbol{c} \\ &\ \ \vdots \\ &= \boldsymbol{T}^l\boldsymbol{x}^{(0)} + \big[\boldsymbol{T}^{(l-1)} + \cdots + \boldsymbol{T}^2 + \boldsymbol{T} + \boldsymbol{I}\big]\boldsymbol{c}. \end{aligned} \tag{2.35}$$

Using equation (2.35), along with Theorem 2.5.6 and the assumption (2.34), then we have the following

$$\lim_{l\to\infty}\left\{\boldsymbol{x}^{(l)}\right\} = \lim_{l\to\infty}\left\{\sum_{k=0}^{l-1}\boldsymbol{T}^k\boldsymbol{c}\right\}. \tag{2.36}$$

From Lemma 2.5.5, we know (2.36) is equivalent to

$$\big[\boldsymbol{I} - \boldsymbol{T}\big]^{-1}\boldsymbol{c}.$$

Therefore the sequence $\{\boldsymbol{x}^{(l)}\}_{l=0}^{\infty}$ converges to the unique solution (2.33) since

$$\boldsymbol{x} = \big[\boldsymbol{I} - \boldsymbol{T}\big]^{-1}\boldsymbol{c}.$$

On the other hand, let us assume (2.33) possesses the unique solution denoted by $\boldsymbol{x}^*$. Now if we also introduce an arbitrary vector

$$\boldsymbol{y} \in \mathbb{R}^n$$

and take the initial guess

$$\boldsymbol{x}^{(0)} = \boldsymbol{x}^*\boldsymbol{y}$$

then we have

$$\begin{aligned}\lim_{l\to\infty}\Big\{\boldsymbol{T}^l\boldsymbol{y}\Big\} &= \lim_{l\to\infty}\Big\{\boldsymbol{T}^l\big[\boldsymbol{x}^* - \boldsymbol{x}^{(0)}\big]\Big\}\\ &= \lim_{l\to\infty}\Big\{\boldsymbol{T}^{l-1}\big[\boldsymbol{x}^* - \boldsymbol{x}^{(1)}\big]\Big\}\\ &\vdots\\ &= \lim_{l\to\infty}\Big\{\big[\boldsymbol{x}^* - \boldsymbol{x}^{(l)}\big]\Big\}.\end{aligned}$$

Since we chose the vector $\boldsymbol{y}$ to be arbitrary, then this implies the matrix $\boldsymbol{T}$ must be convergent, and from Theorem 2.5.4 we get

$$\rho\big(\boldsymbol{T}\big) < 1.$$

□

For a general iterative scheme, we can define the convergence rate by the following

$$\rho = \lim_{l\to\infty}\left\{\left(\sup_{\boldsymbol{e}^{(0)}\in\mathbb{R}^n}\frac{\big\|\boldsymbol{e}^{(l)}\big\|}{\big\|\boldsymbol{e}^{(0)}\big\|}\right)^{\frac{1}{l}}\right\} \tag{2.37}$$

where $\boldsymbol{e}^{(l)}$ denotes the error between the true solution $\boldsymbol{x}^*$ of the system (2.21) and the approximation at step l $\boldsymbol{x}^{(l)}$, which is defined by

$$\boldsymbol{e}^{(l)} = \boldsymbol{x}^* - \boldsymbol{x}^{(l)}. \tag{2.38}$$

From (2.38), we can write

$$\begin{aligned}
\boldsymbol{e}^{(l)} &= \boldsymbol{x}^* - \boldsymbol{x}^{(l)} \\
&= \boldsymbol{T}\boldsymbol{x}^* + \boldsymbol{c} - \left[\boldsymbol{T}\boldsymbol{x}^{(l-1)} + \boldsymbol{c}\right] \\
&= \boldsymbol{T}\left[\boldsymbol{x}^* - \boldsymbol{x}^{(l-1)}\right] \\
&= \boldsymbol{T}\boldsymbol{e}^{(l-1)} \\
&\vdots \\
&= \boldsymbol{T}^l \boldsymbol{e}^{(0)}.
\end{aligned} \tag{2.39}$$

Then using (2.39), and the fact for any matrix norm

$$\lim_{l\to\infty} \left\{ \left(\left\| \boldsymbol{M}^l \right\|^{\frac{1}{l}} \right) \right\} = \rho(\boldsymbol{M})$$

we can write the expression for the convergence rate (2.37) in the following way

$$\rho = \lim_{l\to\infty} \left\{ \left(\left\| \boldsymbol{T}^l \right\| \right)^{\frac{1}{l}} \right\} = \rho(\boldsymbol{T})$$

and thus we see the optimal iterative method is the one whose iteration matrix possesses the smallest spectral radius.

Now we present some useful convergence theorems [84] which we state without proof.

Theorem 2.5.7 (Spectral radius of the Jacobi and Gauss-Seidel methods)**.** *If a matrix* $\boldsymbol{A}$ *has positive diagonal entries and all other entries are negative or zero, then only one of the following statements hold:*

(i) $0 < \rho(\boldsymbol{T}_{GS}) < \rho(\boldsymbol{T}_J) < 1$;

(ii) $1 < \rho(\boldsymbol{T}_J) < \rho(\boldsymbol{T}_{GS})$;

(iii) $\rho(\boldsymbol{T}_J) = \rho(\boldsymbol{T}_{GS}) = 0$;

(iv) $\rho(\boldsymbol{T}_J) = \rho(\boldsymbol{T}_{GS}) = 1$;

where $\boldsymbol{T}_J$ *and* $\boldsymbol{T}_{GS}$ *denote the iteration matrices, as shown in* (2.25) *and* (2.30)*, for the Jacobi and GS iterative methods respectively.*

From Theorem 2.5.7, we see if either the Jacobi or GS method converges then so does the other. Conversely, if one of the two diverges, then the other does too. Moreover, if the two methods converge, then the GS method will converge faster than the Jacobi method.

Remark 2.5.8. *In order for the Jacobi method to converge, a sufficient condition is the system matrix* $\boldsymbol{A}$ *must be strictly diagonally dominant. For convergence of the Gauss-Seidel method however, one of the two following conditions must be satisfied:*

(i) The system matrix $\boldsymbol{A}$ *is strictly diagonally dominant;*

(ii) The system matrix $\boldsymbol{A}$ *is symmetric positive definite (SPD).*

All image registration models considered in this thesis possess diagonally dominant SPD system matrices, hence the Jacobi and Gauss-Seidel methods will converge for these models.

Definition 2.5.9 (Regular splitting)**.**

$$\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}$$

is called a regular splitting if $\boldsymbol{M}$ *is non-singular and* $\boldsymbol{M}$, $\boldsymbol{N}$ *are non-negative.*

Theorem 2.5.10. *If* $\boldsymbol{M}$ *and* $\boldsymbol{N}$ *are a regular splitting of* $\boldsymbol{A}$*, and*

$$\boldsymbol{T} = \boldsymbol{M}^{-1}\boldsymbol{N}$$

then $\rho(\boldsymbol{T}) < 1 \iff \boldsymbol{A}$ *is non-singular and* $\boldsymbol{A}^{-1}$ *non-negative.*

Theorem 2.5.11. *If all of the diagonal elements of* $\boldsymbol{A}$ *are non-zero then*

$$\rho(\boldsymbol{T}_{SOR}) \geq |\omega - 1|$$

hence the SOR method converges only when $0 < \omega < 2$*.*

Theorem 2.5.12. *If* $\boldsymbol{A}$ *is a positive definite matrix, i.e.*

$$\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} > 0$$

for any $\boldsymbol{x}$*, and* $0 < \omega < 2$*, then the SOR method converges for any initial guess* $\boldsymbol{x}^{(0)}$*.*

Theorem 2.5.13. *If the matrix* $\boldsymbol{A}$ *is positive definite and tri-diagonal, then*

$$\rho(\boldsymbol{T}_{GS}) = \rho(\boldsymbol{T}_J)^2$$

and the optimal value for ω *is*

$$\omega = \frac{2}{1 + \sqrt{1 - \rho(\boldsymbol{T}_J)^2}}$$

for which

$$\rho(\boldsymbol{T}_{SOR}) = \omega - 1.$$

2.5.6 Iterative methods for non-linear equations

Suppose we now wish to solve the following non-linear system

$$F_i\big(x_1,\ldots,x_n\big) = 0 \text{ for } i = 1,\ldots,n \tag{2.40}$$

which we can obtain from the discretisation of a non-linear PDE or optimisation problem of the following form

$$\min\big\{E\big(x_1,\ldots,x_n\big)\big\}.$$

Then the system (2.40) can be written in the form

$$\boldsymbol{F}(\boldsymbol{x}) = \boldsymbol{0},\ \boldsymbol{F} = \big[F_1,\ldots,F_n\big]^T,\ \boldsymbol{x} = \big[x_1,\ldots,x_n\big]^T \tag{2.41}$$

and where $\boldsymbol{0}$ denotes the zero column vector of dimension $n \times 1$. Our aim is to find the solution $\boldsymbol{x}^* \in \mathbb{R}^n$ of the system (2.40). For the remainder of this section, we outline several methods which we can use to solve non-linear systems such as the one shown in (2.40).

2.5.7 Newton method

Let us begin by denoting the Jacobian matrix of $\boldsymbol{F}$ by $\boldsymbol{J}$, and define it in the following way

$$\boldsymbol{J} = \big(J\big)_{ij} = \frac{\partial F_i(\boldsymbol{x})}{\partial x_j}$$

for $i,\, j = 1,\ldots,n$. In addition let us assume the Jacobian $\boldsymbol{J}$ is Lipschitz continuous, then the Newton method looks to find the solution of (2.41) via the use of the following recurrence relation

$$\boldsymbol{x}^{(l)} = \boldsymbol{x}^{(l-1)} - \Big[\boldsymbol{J}(\boldsymbol{x}^{(l-1)})\Big]^{-1}\boldsymbol{F}(\boldsymbol{x}^{(l-1)})$$

which can be written as

$$\textit{Solve } \boldsymbol{d}^{(l-1)} = -\Big[\boldsymbol{J}(\boldsymbol{x}^{(l-1)})\Big]^{-1}\boldsymbol{F}(\boldsymbol{x}^{(l-1)}),$$

$$\textit{Update } \boldsymbol{x}^{(l)} = \boldsymbol{x}^{(l-1)} + \boldsymbol{d}^{(l-1)}$$

where $\boldsymbol{d}^{(l-1)}$ denotes the search direction.

2.5.8 Gradient descent method

Suppose we seek to find the solution $\boldsymbol{x}$ of the non-linear system (2.41), then we can use a method called the gradient (or steepest) descent method. This method generates a sequence $\boldsymbol{x}^{(l)}$, for $l \geq 1$, according to the following recurrence relation

$$\boldsymbol{x}^{(l)} = \boldsymbol{x}^{(l-1)} + \alpha^{(l-1)}\boldsymbol{d}^{(l-1)},\ \boldsymbol{d}^{(l-1)} = -\nabla \boldsymbol{F}(\boldsymbol{x}^{(l-1)}) \tag{2.42}$$

where $\alpha^{(l-1)}$ denotes the so-called step length and $\boldsymbol{d}^{(l-1)}$ the search direction. The purpose of the gradient descent method is the search direction $\boldsymbol{d}^{(l-1)}$ is always a descent direction, and as a result each iteration decreases from the previous. In other words we have

$$\boldsymbol{F}(\boldsymbol{x}^{(l)}) \leq \boldsymbol{F}(\boldsymbol{x}^{(l-1)}).$$

Now supposing we replace the step length $\alpha^{(l-1)}$ with the time step Δt in (2.42), then we obtain the so-called time-marching method [47–49, 72, 74, 90, 100, 126].

2.5.9 Quasi-Newton method

Let us suppose we have an optimisation problem of the following form

$$\min_{\boldsymbol{x}} \big\{E(\boldsymbol{x})\big\} \tag{2.43}$$

where E is some functional and

$$E\colon \mathbb{R}^n \to \mathbb{R} \in C^2.$$

Also suppose we are given some initial guess $\boldsymbol{x}^{(0)}$, then we look to introduce an iterative method to generate a sequence of approximations which converge to the true minimum value $\boldsymbol{x}^*$. Moreover, let us denote the gradient of E at $\boldsymbol{x}^{(l)}$ by

$$\nabla E(\boldsymbol{x}^{(l)})$$

and the Hessian matrix by

$$\boldsymbol{H}^{(l)} = \nabla^2 E(\boldsymbol{x}^{(l)}).$$

The second order Taylor expansion of the functional E, around the approximation $\boldsymbol{x}^{(l)}$, is defined by the following

$$\hat{E}(\boldsymbol{\varepsilon}) = E(\boldsymbol{x}^{(l)}) + \boldsymbol{\varepsilon}^T \nabla E(\boldsymbol{x}^{(l)}) + \frac{1}{2}\boldsymbol{\varepsilon}^T \boldsymbol{H}^{(l)} \boldsymbol{\varepsilon}$$

where

$$\boldsymbol{\varepsilon} = \boldsymbol{x} - \boldsymbol{x}^{(l)}.$$

Now the expression $\hat{E}(\boldsymbol{\varepsilon})$ defines a quadratic model of E around $\boldsymbol{x}^{(l)}$, and the gradient with respect to $\boldsymbol{x}$ is defined by the following

$$\nabla \hat{E}(\boldsymbol{\varepsilon}) = \nabla E(\boldsymbol{x}^{(l)}) + \boldsymbol{H}^{(l)}\boldsymbol{\varepsilon}. \tag{2.44}$$

For minimal values of the expression $\nabla \hat{E}(\boldsymbol{\varepsilon})$ in (2.44), we require the following

$$\boldsymbol{\varepsilon}^{(l)} = -\left[\boldsymbol{H}^{(l)}\right]^{-1} \nabla E(\boldsymbol{x}^{(l)}) \tag{2.45}$$

and so the Newton method is given by the following recurrence relation

$$\boldsymbol{x}^{(l+1)} = \boldsymbol{x}^{(l)} + \alpha^{(l)}\boldsymbol{\varepsilon}^{(l)} \tag{2.46}$$

where the value of $\boldsymbol{\varepsilon}^{(l)}$ is determined by solving (2.45). However, since the solution of (2.45) requires the computation of the inverse of the Hessian matrix, i.e.

$$\left[\boldsymbol{H}^{(l)}\right]^{-1}$$

this method may be too expensive computationally for large problems. In order to avoid this cost, and directly computing the inverse Hessian matrix, the quasi-Newton method seeks to find an approximation of the Hessian matrix which is easier to invert. Some examples of how the Hessian can be approximated can be seen in the works [98, 127].

2.5.10 Line search method

For recurrence relations of the form shown in (2.46) the choice of the step length $\alpha^{(l)}$, in addition to the search direction $\boldsymbol{\varepsilon}^{(l)}$, need to be carefully considered in order to obtain a method which is convergent. While the initialisation of the step length is typically taken to be $\alpha^{(0)} = 1$, for subsequent iterations we can impose the following condition in order to get a reduction in E

$$E(\boldsymbol{x}^{(l)} + \alpha^{(l)}\boldsymbol{\varepsilon}^{(l)}) \leq E(\boldsymbol{x}^{(l)}).$$

However, as can be seen in [144], this condition is not sufficient to guarantee convergence of the solution. A popular alternate condition is the so-called Wolfe condition [142, 143], which is given by the following

$$E(\boldsymbol{x}^{(l)} + \alpha^{(l)}\boldsymbol{\varepsilon}^{(l)}) \leq E(\boldsymbol{x}^{(l)}) + c\alpha^{(l)}\nabla E(\boldsymbol{x}^{(l)})^T \varepsilon^{(l)}$$

where $0 < c < 1$ denotes some constant.

2.6 Multigrid methods

Multigrid (or MG) methods are a form of multilevel strategy first proposed by A. Brandt in [9]. They were developed to be efficient solvers of a large selection of both linear and non-linear discrete elliptic PDEs. In MG methods, the idea is to smooth out any high frequency components of the solution error in the Fourier domain through the use of a few iterations of a given 'smoother' scheme (such as the iterative methods outlined in §2.5). Once the error has been sufficiently smoothed, we restrict the problem onto a coarser grid, where we solve a linear/non-linear residual equation. With this accurate solution, we can compute the so-called coarse grid correction which can then be interpolated back to the fine grid and used to update the approximation on the fine grid. After the approximation has been corrected, we perform another smoother step to remove any high frequency interpolation errors which may have been introduced. This scheme is known as the two-grid V-cycle, for more details on the introduction to MG methods see [10, 25, 131].

Definition 2.6.1 (Fourier mode)**.** *Given some initial vector*

$$\boldsymbol{u}^{(0)} = \left(u_1^{(0)}, \ldots, u_n^{(0)}\right)$$

then the Fourier mode of $u_i^{(0)}$ *is defined by the following*

$$u_i^{(0)} = \sin\left(\frac{ik\pi}{n}\right)$$

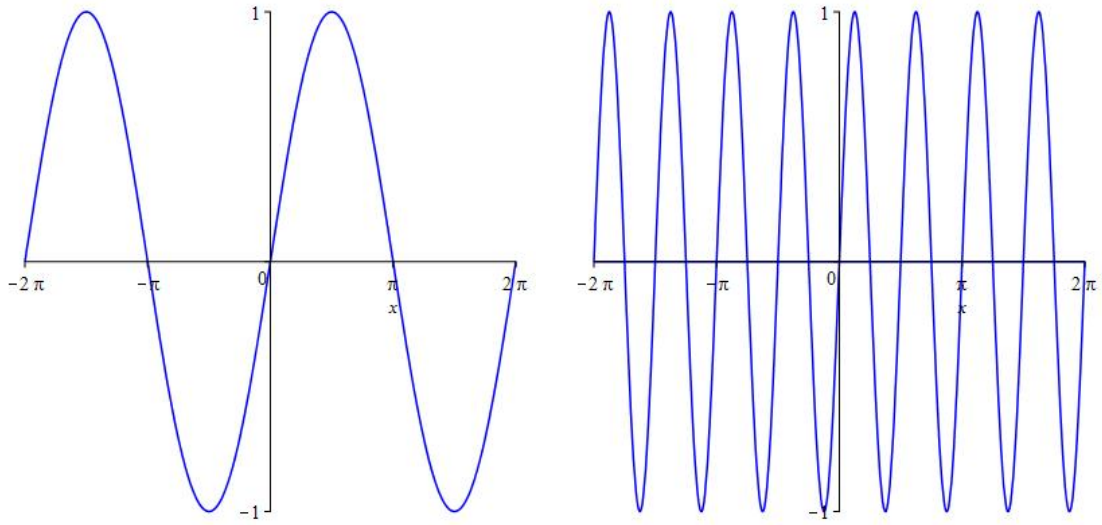
where $0 \leq i \leq n$ *and* $1 \leq k \leq n-1$ *denotes the wavenumber or the frequency of* $\boldsymbol{u}^{(0)}$*. Note small values of* k *result in a vector* $\boldsymbol{u}^{(0)}$ *with low oscillations, while a high value results in a vector* $\boldsymbol{u}^{(0)}$ *with high oscillations. An example of low and high frequency oscillations can be seen in Figure 2.5.*

2.6.1 The basic principles of multigrid

MG methods are based upon two key ideas, namely the principles of smoothing and coarsening.

Smoothing Principle. In general, relaxation schemes like the ones discussed in §2.5, can be very slow to converge when applied to discrete elliptic PDEs. However, these same schemes are effective at removing high frequency Fourier components (Fourier modes with large k values in Definition 2.6.1 and seen in Figure 2.5(b)), and so these techniques are very effective for use in the smoother steps of the MG method. This then leads into the second key MG principle.

Coarsening Principle. According to the Nyquist-Shannon theorem [122] only low frequency components of the fine grid error can be well approximated on a coarser grid.



(a) Low frequency oscillations corresponding to $k = 1$.

(b) High frequency oscillations corresponding to $k = 4$.

Figure 2.5: Visual representation of low frequency and high frequency oscillations.

Let us consider the following linear system

$$\boldsymbol{A}\boldsymbol{u} = \boldsymbol{f}. \tag{2.47}$$

Moreover let $\tilde{\boldsymbol{u}}$ be an approximation of the solution $\boldsymbol{u}$, then we can define the solution error by the following

$$\boldsymbol{e} = \boldsymbol{u} - \tilde{\boldsymbol{u}}. \tag{2.48}$$

By applying the system matrix $\boldsymbol{A}$ to both sides of the error equation (2.48), we can obtain the so-called residual (or defect) equation which is given by the following

$$\boldsymbol{A}\boldsymbol{e} = \boldsymbol{A}\big(\boldsymbol{u} - \tilde{\boldsymbol{u}}\big) = \boldsymbol{F} - \boldsymbol{A}\tilde{\boldsymbol{u}} \equiv \boldsymbol{r} \tag{2.49}$$

where $\boldsymbol{r}$ denotes the residual. From (2.49) we see it is possible to obtain $\boldsymbol{u}$ using

$$\boldsymbol{u} = \tilde{\boldsymbol{u}} + \boldsymbol{e}.$$

However, this assumes we can solve (2.49) exactly which may be just as expensive as solving the system (2.47). Instead we look to approximate $\boldsymbol{A}$ on a coarser grid where it is much cheaper to find the error $\boldsymbol{e}$.

To outline the MG method, let us consider the following discrete linear system

$$\boldsymbol{\mathcal{L}}^h \boldsymbol{u}^h = \boldsymbol{f}^h$$

which results from some elliptic PDE on the discrete domain Ω^h with grid spacing

$$\boldsymbol{h} = (h_1, h_2)\,.$$

Also let $\tilde{\boldsymbol{u}}^h$ denote the smooth approximation obtained from the pre-smoothing step of the fine grid problem. Then we define the residual equation by the following

$$\boldsymbol{\mathcal{L}}^h \boldsymbol{e}^h = \boldsymbol{f}^h - \boldsymbol{\mathcal{L}}^h \tilde{\boldsymbol{u}}^h \equiv \boldsymbol{r}^h \tag{2.50}$$

where we define the error

$$\boldsymbol{e}^h = \boldsymbol{u}^h - \tilde{\boldsymbol{u}}^h$$

and where $\boldsymbol{r}^h$ denotes the fine grid residual. Since only low frequency error components are left after the pre-smoothing step, we can transfer the fine grid residual equation (2.50) to a coarser grid which we denote by Ω^H with spacing

$$\boldsymbol{H} = (H_1, H_2)\,.$$

Doing so gives us the following coarse grid residual equation

$$\boldsymbol{\mathcal{L}}^h \boldsymbol{e}^h = \boldsymbol{r}^h \longrightarrow \boldsymbol{\mathcal{L}}^H \boldsymbol{e}^H = \mathcal{R}_h^H \boldsymbol{r}^h \equiv \boldsymbol{r}^H \tag{2.51}$$

where we assume the operator $\boldsymbol{\mathcal{L}}^H$ is an appropriate coarse grid approximation of the fine grid operator $\boldsymbol{\mathcal{L}}^h$, and $\mathcal{R}_h^H$ denotes the restriction operator used to transfer quantities between the fine grid Ω^h and the coarse grid Ω^H (see §2.6.3). The coarse grid residual equation (2.51) is then solved exactly, using some chosen method, thus allowing us to determine the coarse grid correction $\boldsymbol{e}^H$. Once $\boldsymbol{e}^H$ has been computed, we then use an interpolation step to obtain the fine grid correction $\boldsymbol{e}^h$. In other words we compute

$$\boldsymbol{e}^h = \mathcal{I}_H^h \boldsymbol{e}^H$$

where $\mathcal{I}_H^h$ denotes the interpolation operator used to transfer quantities from the coarse grid Ω^H to the fine grid Ω^h (see §2.6.3). With the fine grid correction $\boldsymbol{e}^h$, we can update the fine grid approximation $\tilde{\boldsymbol{u}}^h$ via

$$\tilde{\boldsymbol{u}}^h_{new} = \tilde{\boldsymbol{u}}^h + \boldsymbol{e}^h.$$

Finally, we perform a post-smoothing step to remove any high frequency error components which may have been introduced by the interpolation. Thus the two-grid V-cycle MG scheme can be summarised by Algorithm 2.

Algorithm 2 $\boldsymbol{u}_h^{(k+1)} \leftarrow Vcycle(\boldsymbol{u}_h^{(k)}, \boldsymbol{\mathcal{L}}^h, \boldsymbol{F}^h, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 relaxation sweeps

$$\tilde{\boldsymbol{u}}_h^{(k)} \leftarrow Smooth(\boldsymbol{u}_h^{(k)}, \boldsymbol{\mathcal{L}}^h, \boldsymbol{F}^h, \nu_1)$$

2: Coarse grid correction

Compute the residual $\boldsymbol{r}_h^{(k)} = \boldsymbol{F}^h - \boldsymbol{\mathcal{L}}^h(\tilde{\boldsymbol{u}}_h^{(k)})$

Restrict residual and smooth approximations $\boldsymbol{r}_H^{(k)} = \mathcal{R}_h^H \boldsymbol{r}_h^{(k)}$, $\tilde{\boldsymbol{u}}_H^{(k)} = \mathcal{R}_h^H \tilde{\boldsymbol{u}}_h^{(k)}$

Set coarse grid interval $\boldsymbol{H}$

Form coarse grid residual equation $\boldsymbol{\mathcal{L}}^H \tilde{\boldsymbol{u}}_H^{(k)} = \boldsymbol{r}_H^{(k)}$

Solve residual equation using a direct or fast iterative solver to obtain accurate solution $\boldsymbol{u}_H^{(k)}$

Compute the correction $\boldsymbol{e}_H^{(k)} = \boldsymbol{u}_H^{(k)} - \tilde{\boldsymbol{u}}_H^{(k)}$

Interpolate the correction to the fine grid level $\boldsymbol{e}_h^{(k)} = \mathcal{I}_H^h \boldsymbol{e}_H^{(k)}$

Update fine grid level approximations using correction $\hat{\boldsymbol{u}}_h^{(k)} = \tilde{\boldsymbol{u}}_h^{(k)} + \boldsymbol{e}_h^{(k)}$

3: Post-smoothing step by performing ν_2 relaxation sweeps

$$\boldsymbol{u}_h^{(k+1)} \leftarrow Smooth(\hat{\boldsymbol{u}}_h^{(k)}, \boldsymbol{\mathcal{L}}^h, \boldsymbol{F}^H, \nu_2)$$

2.6.2 Coarsening

A key part of the MG method is the restriction of the problem onto a coarser grid. In this section we briefly outline what we mean by a coarse grid, in addition to how we transfer our problem to the coarse grid. First let us begin by assuming that we have a discrete Cartesian grid which we denote by Ω^h with grid spacing

$$\boldsymbol{h} = (h_1, h_2)$$

called the fine grid, then we construct the coarse grid denoted by Ω^{2h} with spacing

$$\boldsymbol{H} = (H_1, H_2)\,.$$

Now there are several ways of determining how the coarse grid, and the associated spacing $\boldsymbol{H}$, are constructed depending on the coarsening strategy used which we now give examples of.

Standard Coarsening. Standard coarsening is the simplest, and most widely used, strategy for constructing the coarse grid Ω^{2h}. For this method we simply double the grid spacing in each dimension, on other words we get

$$\boldsymbol{H} = (2h_1, 2h_2)\,.$$

For vertex-centred grids we obtain a coarse grid Ω^{2h} of dimension

$$\left(\frac{n_1}{2} + 1, \frac{n_2}{2} + 1\right)$$

assuming the fine grid is of dimension

$$(n_1 + 1, n_2 + 1)\,.$$

Semi-Coarsening. An alternative to the standard coarsening strategy is to double the grid spacing along only a single dimension, this is known as semi-coarsening. For example, we could have

$$\boldsymbol{H} = (2h_1, h_2) \text{ or } \boldsymbol{H} = (h_1, 2h_2)\,.$$

Such coarsening strategies are typically used in anisotropic problems where smoothers only smooth errors along a single direction.

2.6.3 Transfer operators

From the previous sections we highlighted along with the smoothing step, the process of transferring values from the fine grid Ω^h to the coarse grid Ω^{2h} and vice versa, is another important part of the MG method. To transfer operators from

$$\Omega^h \to \Omega^{2h}$$

we require a restriction operator (which we denote by $\mathcal{R}_h^{2h}$), and to transfer from

$$\Omega^{2h} \to \Omega^h$$

we require an interpolation (or prolongation) operator (which we denote by $\mathcal{I}_{2h}^h$). For the following we only consider transfer operators for standard coarsening and vertex-centred grids, however similar operators can be constructed for the semi-coarsening and cell-centred grid cases.

Restriction operators for vertex-centred grids

In practice there are three possible choices for the restriction operator, these are:

(i) Bijection;

(ii) Half-weighted restriction;

(iii) Full-weighted restriction.

We now briefly describe each of these three choices.

Bijection. Bijection is the simplest, and most intuitive, choice of restriction operator. This type of restriction simply takes every other vertex in each direction, or in other words we have

$$\left(u^{2h}\right)_{i,j} = \left(u^h\right)_{2i,2j}.$$

Half-weighted restriction. The half-weighted restriction operator works by taking a weighted average of five points, and is define by the following

$$\left(u^{2h}\right)_{i,j} = \frac{1}{8}\Big[\left(u^h\right)_{2i,2j-1} + \left(u^h\right)_{2i-1,2j} + 4\left(u^h\right)_{2i,2j} + \left(u^h\right)_{2i+1,2j} + \left(u^h\right)_{2i,2j+1}\Big].$$

Or equivalently, in operator form, we have

$$u^{2h} = \mathcal{R}_h^{2h} u^h \tag{2.52}$$

where we have written the half-weighted restriction operator $\mathcal{R}_h^{2h}$ using the following stencil notation

$$\mathcal{R}_h^{2h} = \frac{1}{8}\begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix}_h^{2h}.$$

Full-weighted restriction. Similar to the half-weighted restriction operator, the full-weighted restriction operator also works by taking a weighted average, however nine points are used instead of five. We define the full-weighted restriction operator by the following

$$\begin{aligned}\left(u^{2h}\right)_{i,j} = \frac{1}{16}\Big[&\left(u^h\right)_{2i-1,2j-1} + \left(u^h\right)_{2i+1,2j-1} + \left(u^h\right)_{2i-1,2j+1} + \left(u^h\right)_{2i+1,2j+1} \\ &+ 2\Big[\left(u^h\right)_{2i,2j-1} + \left(u^h\right)_{2i-1,2j} + \left(u^h\right)_{2i+1,2j} + \left(u^h\right)_{2i,2j+1}\Big] + 4\left(u^h\right)_{2i,2j}\Big].\end{aligned}$$

Again, we can write the full-weighted restriction operator in the operator form (2.52) using the following stencil

$$\mathcal{R}_h^{2h} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^{2h}.$$

Interpolation operator for vertex-centred grids

The most common choice for the interpolation operator is bilinear interpolation, which we define by the following

$$\begin{aligned}\left(u^h\right)_{2i,2j} &= \left(u^{2h}\right)_{i,j}; \\ \left(u^h\right)_{2i+1,2j} &= \frac{1}{2}\Big[\left(u^{2h}\right)_{i,j} + \left(u^{2h}\right)_{i+1,j}\Big]; \\ \left(u^h\right)_{2i,2j+1} &= \frac{1}{2}\Big[\left(u^{2h}\right)_{i,j} + \left(u^{2h}\right)_{i,j+1}\Big]; \\ \left(u^h\right)_{2i+1,2j+1} &= \frac{1}{4}\Big[\left(u^{2h}\right)_{i,j} + \left(u^{2h}\right)_{i+1,j} + \left(u^{2h}\right)_{i,j+1} + \left(u^{2h}\right)_{i+1,j+1}\Big]\end{aligned}$$

and can be written in the following operator form

$$u^h = \mathcal{I}_{2h}^h u^{2h}$$

where the interpolation operator $\mathcal{I}_{2h}^{2h}$ is given by the following stencil

$$\mathcal{I}_{2h}^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_{2h}^{h}.$$

Remark 2.6.2. *We say an interpolation operator has order k if it can precisely transfer polynomials of order $k-1$. In [71] it was explained the sum of the orders of the restriction and interpolation operators must be greater than or equal to the order of the PDE trying to be solved for a MG method to be convergent.*

2.6.4 Local Fourier analysis (LFA)

As we mentioned in §2.6.1, it is very important to smooth out any high frequency error components before we restrict to a coarser grid. For this reason being able to measure how effective a given smoother scheme is at removing high frequency components is crucial. This task can be achieved using a technique called local Fourier analysis or LFA. In LFA we consider how discrete linear operators $\boldsymbol{\mathcal{L}}^h$, with constant coefficients, act upon grid functions which are characterised by

$$\boldsymbol{\varphi}^h(\boldsymbol{x}, \boldsymbol{\theta}) = \exp\left(\frac{\boldsymbol{i\theta}\cdot\boldsymbol{x}}{\boldsymbol{h}}\right) = \exp\left(\frac{\boldsymbol{i}\theta_1 x_1}{h_1} + \frac{\boldsymbol{i}\theta_2 x_2}{h_2}\right)$$

over an infinite grid defined by

$$\Omega_\infty^h = \left\{\boldsymbol{x} = \left(x_{1_i}, x_{2_j}\right) = \left(ih_1, jh_2\right) \colon \left(i, j\right) \in \mathbb{Z}^2\right\}$$

where

$$\boldsymbol{h} = \left(h_1, h_2\right) = \left(\frac{1}{n_1 - 1}, \frac{1}{n_2 - 1}\right)$$

denotes the grid spacing for a vertex-centred grid, $\boldsymbol{i} = \sqrt{-1}$ and $\boldsymbol{\theta} = \left(\theta_1, \theta_2\right)$ denotes the frequency. Now assuming $\boldsymbol{\theta}$ varies continuously in $\mathbb{R}^2$, then it follows

$$\boldsymbol{\varphi}^h\left(\boldsymbol{x}, \boldsymbol{\theta}\right) = \boldsymbol{\varphi}^h\left(\boldsymbol{x}, \bar{\boldsymbol{\theta}}\right),\ \boldsymbol{x} \in \Omega_\infty^h$$

where θ_1, $\bar{\theta_1}$ and θ_2, $\bar{\theta_2}$ differ by multiples of 2π. Owing to the fact the grid functions $\boldsymbol{\varphi}^h\left(\boldsymbol{x}, \boldsymbol{\theta}\right)$ are periodic, then we need only consider the range

$$\boldsymbol{\theta} = [-\pi, \pi)^2 \equiv \boldsymbol{\Theta}$$

(see [131]). We now define low frequency components by the grid functions $\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$ with frequency

$$\boldsymbol{\theta} \in \boldsymbol{\Theta}_{low} = \Big[-\frac{\pi}{2},\frac{\pi}{2}\Big)^2$$

and high frequency components by the grid functions $\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$ with frequency

$$\boldsymbol{\theta} \in \boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \setminus \boldsymbol{\Theta}_{low}.$$

Theorem 2.6.3. *For $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, all grid functions $\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$ are eigenfunctions of any discrete linear operator $\boldsymbol{\mathcal{L}}^h$ with constant coefficients and the following relation holds*

$$\boldsymbol{\mathcal{L}}^h\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$$

where $\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the linear operator $\boldsymbol{\mathcal{L}}^h$, and is defined by

$$\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta}) = \sum_{\boldsymbol{p}\in\mathbb{Z}^2} \boldsymbol{L}_{\boldsymbol{p}} e^{\boldsymbol{i}\boldsymbol{\theta}\cdot\boldsymbol{p}}.$$

Proof. A proof of Theorem 2.6.3 can be seen in [131]. □

We can now use Theorem 2.6.3 to perform an analysis on the smoothing properties of a given smoother scheme used to solve a discrete PDE which we denote by

$$\boldsymbol{\mathcal{L}}^h\boldsymbol{u}^h = \boldsymbol{F}^h. \tag{2.53}$$

First we use the assumption we can write a single step of the smoother scheme locally in the following way

$$\boldsymbol{\mathcal{L}}^h_+\boldsymbol{u}^h_{new} + \boldsymbol{\mathcal{L}}^h_-\boldsymbol{u}^h_{old} = \boldsymbol{F}^h \tag{2.54}$$

where we have denoted the current and previous approximations of $\boldsymbol{u}^h$ by $\boldsymbol{u}^h_{new}$ and $\boldsymbol{u}^h_{old}$ respectively, and also where we have split the discrete linear operator $\boldsymbol{\mathcal{L}}^h$ in the following way

$$\boldsymbol{\mathcal{L}}^h = \boldsymbol{\mathcal{L}}^h_+ + \boldsymbol{\mathcal{L}}^h_-. \tag{2.55}$$

Then subtracting (2.54) from the original discrete PDE (2.53), we can obtain the following local error equations

$$\boldsymbol{\mathcal{L}}^h_+\boldsymbol{e}^h_{new} + \boldsymbol{\mathcal{L}}^h_-\boldsymbol{e}^h_{old} = 0$$

which we can rearrange to get the following

$$\boldsymbol{e}^h_{new} = -\left[\boldsymbol{\mathcal{L}}^h_+\right]^{-1}\boldsymbol{\mathcal{L}}^h_-\boldsymbol{e}^h_{old} \equiv \boldsymbol{\mathcal{S}}^h\boldsymbol{e}^h_{old}.$$

Using (2.55), along with Theorem 2.6.3, we notice the grid functions $\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$ are eigenfunctions of $\boldsymbol{\mathcal{S}}^h$, therefore we have

$$\boldsymbol{\mathcal{S}}^h\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{S}}}^h(\boldsymbol{\theta})\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}^h_+(\boldsymbol{\theta})\right]^{-1}\hat{\boldsymbol{\mathcal{L}}}^h_-(\boldsymbol{\theta})\boldsymbol{\varphi}^h(\boldsymbol{x},\boldsymbol{\theta})$$

under the assumption

$$\hat{\boldsymbol{\mathcal{L}}}^h_+(\boldsymbol{\theta}) \neq 0.$$

Then we define the so-called local smoothing rate of a given smoother scheme by the following

$$\mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\left|\hat{\boldsymbol{\mathcal{S}}}^h(\boldsymbol{\theta})\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}.$$

For a smoother scheme to remove any high frequency error components we require $\mu_{loc} < 1$, and the smaller the value of μ_{loc} is the better the smoother scheme is at removing these components and the fewer iterations which will be required in the smoothing step.

Remark 2.6.4. *While this analysis has only been shown for linear operators, the work done by A. Brandt in [9] allowed this analysis to be extended to work with non-linear operators by locally 'freezing' the coefficients, and thus allowing the non-linear operator to be approximated locally by a linear operator.*

2.6.5 Multigrid cycles

So far we have only explained how the MG method works in the two-grid setting, and while the coarse grid Ω^{2h} possesses four times fewer grid points compared with the fine grid Ω^h, a direct solution to the residual equation is probably still too expensive computationally to perform. Instead we can perform another smoother step on the coarse grid correction, and then solve the residual equation on the even coarser grid Ω^{4h} which has four times fewer grid points than the coarse grid Ω^{2h} and sixteen times fewer than the original fine grid Ω^h. We can keep repeating this process to recursively interact with even more coarse grids, until a coarse enough grid is reached where a direct solution to the residual equation can be computed efficiently. Such a technique is referred to as a μ-cycle MG step if μ coarse grid corrections have been used to solve the residual equation approximately. In practice, only $\mu = 1$ and $\mu = 2$ are used which result in the so-called V-cycle and W-cycle MG methods respectively. The algorithm for the μ-cycle MG method is shown in Algorithm 3, and diagrams of the V-cycle and W-cycle MG methods can be seen in Figure 2.6 for the case of four grid levels.

Algorithm 3 $\boldsymbol{u}_{level}^{(k+1)} \leftarrow \mu cycle(level, \mu, \boldsymbol{u}_{level}^{(k)}, \boldsymbol{\mathcal{L}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 steps

$$\tilde{\boldsymbol{u}}_l^{(k)} \leftarrow Smooth(\boldsymbol{u}_{level}^{(k)}, \boldsymbol{\mathcal{L}}_{level}, \boldsymbol{F}_{level}, \nu_1)$$

2: Coarse grid correction

Compute the residual $\boldsymbol{r}_{level}^{(k)} = \boldsymbol{F}_{level} - \boldsymbol{\mathcal{L}}_{level}(\tilde{\boldsymbol{u}}_l^{(k)})$

Restrict residual and smooth approximations $\boldsymbol{r}_{level-1}^{(k)} = \mathcal{R}_{level}^{level-1}\boldsymbol{r}_{level}^{(k)}$, $\tilde{\boldsymbol{u}}_{level-1}^{(k)} = \mathcal{R}_{level}^{level-1}\tilde{\boldsymbol{u}}_{level}^{(k)}$

Set $level \rightarrow level - 1$

Form coarse grid residual equation $\boldsymbol{\mathcal{L}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(k)} = \boldsymbol{r}_{level-1}^{(k)}$

Solve residual equation on coarse grid to obtain approximations $\bar{\boldsymbol{u}}_{level-1}^{(k)}$

3: **if** $level = 1$ **then**

Solve to obtain solutions $\boldsymbol{u}_{level-1}^{(k)}$ to high accuracy using a direct or fast iterative solver.

4: **else** $level > 1$ Repeat the μ-cycle procedure recursively to the next level using zero grid functions as initial approximation i.e.

$$\tilde{\boldsymbol{u}}_{level}^{(k)} \leftarrow \mu cycle(level - 1, \mu, 0, \boldsymbol{\mathcal{L}}_{level-1}, \boldsymbol{F}_{level-1}, \nu_1, \nu_2)$$

5: **end if**

Compute the correction $\boldsymbol{e}_{level-1}^{(k)} = \boldsymbol{u}_{level-1}^{(k)} - \tilde{\boldsymbol{u}}_{level-1}^{(k)}$

Interpolate the correction to next fine grid level $\boldsymbol{e}_{level}^{(k)} = \mathcal{I}_{level-1}^{level}\boldsymbol{e}_H^{(k)}$

Update current grid level approximations using correction $\hat{\boldsymbol{u}}_{level}^{(k)} = \tilde{\boldsymbol{u}}_{level}^{(k)} + \boldsymbol{e}_{level}^{(k)}$

6: Post-smoothing step by performing ν_2 steps

$$\boldsymbol{u}_{level}^{(k+1)} \leftarrow Smooth(\hat{\boldsymbol{u}}_{level}^{(k)}, \boldsymbol{\mathcal{L}}_{level}, \boldsymbol{F}_{level}, \nu_2)$$

2.6.6 Full multigrid methods

In the full multigrid method, we begin by solving the problem on the coarsest level in order to gain a very good initial guess for the next fine level which we obtain by interpolation. This process is then repeated until we reach the original fine grid level, and a visual representation of this procedure can be seen in Figure 2.6. The algorithm for the full multigrid method is shown in Algorithm 4.

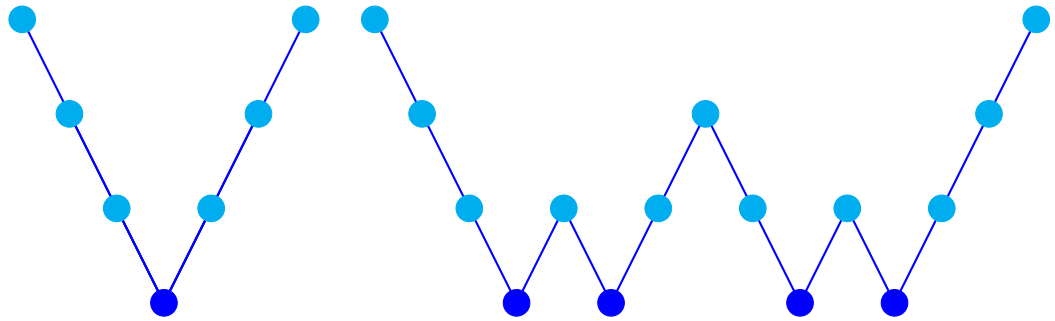
2.6.7 Full approximation scheme non-linear multigrid (FAS-NMG)

The full approximation scheme non-linear multigrid (or FAS-NMG) is a very powerful multigrid technique for solving discrete non-linear PDEs. Let us consider the non-linear PDE

$$\boldsymbol{\mathcal{N}}^h\boldsymbol{u}^h = \boldsymbol{F}^h$$

discretised on the fine grid Ω^h, and where $\boldsymbol{\mathcal{N}}^h$ denotes a non-linear operator acting on $\boldsymbol{u}^h$. Furthermore let us denote the smooth approximation of $\boldsymbol{u}^h$, obtained by performing a few iterations of a smoother scheme (such as the ones described in §2.5) by $\tilde{\boldsymbol{u}}^h$. Then we can define the non-linear fine grid residual equation by the following

$$\boldsymbol{\mathcal{N}}^h\boldsymbol{u}^h - \boldsymbol{\mathcal{N}}^h\tilde{\boldsymbol{u}}^h \equiv \boldsymbol{\mathcal{N}}^h\big(\tilde{\boldsymbol{u}}^h + \boldsymbol{e}^h\big) - \boldsymbol{\mathcal{N}}^h\tilde{\boldsymbol{u}}^h = \boldsymbol{F}^h - \boldsymbol{\mathcal{N}}^h\tilde{\boldsymbol{u}}^h \equiv \boldsymbol{r}^h \tag{2.56}$$



(a) V-cycle multigrid ($\mu = 1$). (b) W-cycle multigrid ($\mu = 2$).

Figure 2.6: Visual representation the V-cycle and W-cycle multigrid methods. Light blue nodes represent smoother steps while dark blue nodes represent an exact solution step, also \ and / correspond to fine-to-coarse restriction and coarse-to-fine interpolation respectively.

where

$$\boldsymbol{e}^h = \boldsymbol{u}^h - \tilde{\boldsymbol{u}}^h$$

denotes the solution error and

$$\boldsymbol{r}^h = \boldsymbol{F}^h - \boldsymbol{\mathcal{N}}^h \tilde{\boldsymbol{u}}^h$$

the non-linear residual. As was the case for the linear MG method, in order to be able to compute the correction $\boldsymbol{e}^h$, we first need to compute the coarse grid correction $\boldsymbol{e}^{2h}$ and interpolate back. Therefore we need to transfer the non-linear residual equation (2.56) to the coarse grid Ω^{2h}. Doing so leads to the following

$$\underbrace{\boldsymbol{\mathcal{N}}^h\big(\tilde{\boldsymbol{u}}^h + \boldsymbol{e}^h\big)}_{\boldsymbol{\mathcal{N}}^h \boldsymbol{u}^h} = \underbrace{\boldsymbol{r}^h + \boldsymbol{\mathcal{N}}^h \tilde{\boldsymbol{u}}^h}_{\boldsymbol{F}^h} \longrightarrow \underbrace{\boldsymbol{\mathcal{N}}^{2h}\big(\tilde{\boldsymbol{u}}^{2h} + \boldsymbol{e}^{2h}\big)}_{\boldsymbol{\mathcal{N}}^{2h} \boldsymbol{u}^{2h}} = \underbrace{\boldsymbol{r}^{2h} + \boldsymbol{\mathcal{N}}^{2h} \tilde{\boldsymbol{u}}^{2h}}_{\boldsymbol{F}^{2h}}. \tag{2.57}$$

Again the next step is similar to the linear case whereby we solve (2.57) using some chosen method to determine $\boldsymbol{u}^{2h}$, and obtain the coarse grid correction

$$\boldsymbol{e}^{2h} = \boldsymbol{u}^{2h} - \tilde{\boldsymbol{u}}^{2h}.$$

Then we interpolate the correction $\boldsymbol{e}^{2h}$ back to Ω^h to get $\boldsymbol{e}^h$ which we can use to update the fine grid approximation via

$$\tilde{\boldsymbol{u}}^h_{new} = \tilde{\boldsymbol{u}}^h + \boldsymbol{e}^h.$$

Finally we require another smoother step to remove any interpolation errors. Naturally we can extend this two-grid case to interact recursively with even coarser grids (like in §2.6.5) to obtain the μ-cycle FAS-NMG method which can be summarised by Algorithm 5.

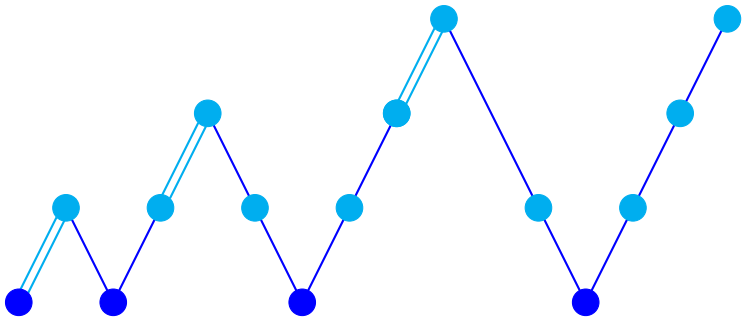


Figure 2.7: Illustration of the full μ-cycle multigrid method with $\mu = 1$. Again light blue nodes represent smoother steps, dark blue nodes represent an exact solutions step and \, / represent restriction, smoother steps respectively. In addition // correspond to FMG interpolation steps.

Algorithm 4 $\boldsymbol{u}_{level}^{(k+1)} \leftarrow FullMG(maxlevel, \boldsymbol{\mathcal{L}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$

1: Coarse grid initialisation
Set $level = 0$
Solve $\boldsymbol{\mathcal{L}}_{level}\boldsymbol{u}_{level} = \boldsymbol{F}_{level}$ to obtain initial guess $\hat{\boldsymbol{u}}_{level}$
2: **for** $level = 1, \ldots, maxlevel$ **do**
Interpolate coarse grid solution $\boldsymbol{u}_{level-1}^{(0)} = \mathcal{I}_{level-1}^{level}\hat{\boldsymbol{u}}_{level}$
Employ μ-cycle multigrid algorithm (Algorithm 3) using $\boldsymbol{u}_{level-1}^{(0)}$ as an initialisation

$$\boldsymbol{u}_{level} = \mu cycle(level + 1, \mu, \boldsymbol{u}_{level}^{(0)}, \boldsymbol{\mathcal{L}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$$

3: **end for**

Algorithm 5 $\boldsymbol{u}_{level}^{(k+1)} \leftarrow FAScycle(level, \mu, \boldsymbol{u}_{level}^{(k)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 steps

$$\tilde{\boldsymbol{u}}_{level}^{(k)} \leftarrow Smooth(\boldsymbol{u}_{level}^{(k)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1)$$

2: Coarse grid correction
Compute the residual $\boldsymbol{r}_{level}^{(k)} = \boldsymbol{F}_{level} - \boldsymbol{\mathcal{N}}_{level}(\tilde{\boldsymbol{u}}_{level}^{(k)})$
Restrict residual and smooth approximations $\boldsymbol{r}_{level-1}^{(k)} = \mathcal{R}_{level}^{level-1}\boldsymbol{r}_{level}^{(k)}$, $\tilde{\boldsymbol{u}}_{level-1}^{(k)} = \mathcal{R}_{level}^{level-1}\tilde{\boldsymbol{u}}_{level}^{(k)}$
Set $level \rightarrow level - 1$
Compute RHS of coarse grid PDE $\boldsymbol{f}_{level-1} = \boldsymbol{r}_{level-1}^{(k)} + \boldsymbol{\mathcal{N}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(k)}$
Compute an approximation $\bar{\boldsymbol{u}}_{level-1}^{(k)}$ to the coarse grid PDE $\boldsymbol{\mathcal{N}}_{level-1}\bar{\boldsymbol{u}}_{level-1}^{(k)} = \boldsymbol{r}_{level-1}^{(k)}$
Solve residual equation on coarse grid to obtain approximations $\bar{\boldsymbol{u}}_{level-1}^{(k)}$
3: **if** $level = 1$ **then**
Use a direct or fast iterative solver to obtain the high accuracy solutions $\boldsymbol{u}_{level-1}^{(k)}$
4: **else** $level > 1$ Repeat the FAS-cycle procedure recursively to the next level using $\tilde{\boldsymbol{u}}_{level-1}^{(k)}$ as an initial approximation i.e.

$$\bar{\boldsymbol{u}}_{level}^{(k)} \leftarrow FAScycle(level - 1, \mu, \tilde{\boldsymbol{u}}_{level-1}^{(k)}, \boldsymbol{\mathcal{N}}_{level-1}, \boldsymbol{F}_{level-1}, \nu_1, \nu_2)$$

5: **end if**
Compute the correction $\boldsymbol{e}_{level-1}^{(k)} = \boldsymbol{u}_{level-1}^{(k)} - \tilde{\boldsymbol{u}}_{level-1}^{(k)}$
Interpolate the correction to next fine grid level $\boldsymbol{e}_{level}^{(k)} = \mathcal{I}_{level-1}^{level}\boldsymbol{e}_{level-1}^{(k)}$
Update current grid level approximations using correction $\hat{\boldsymbol{u}}_{level}^{(k)} = \tilde{\boldsymbol{u}}_{level}^{(k)} + \boldsymbol{e}_{level}^{(k)}$
6: Post-smoothing step by performing ν_2 steps

$$\boldsymbol{u}_{level}^{(k+1)} \leftarrow Smooth(\hat{\boldsymbol{u}}_{level}^{(k)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_2)$$

Chapter 3

Mathematical models for image registration

In this chapter we describe the general framework for image registration, in addition to showing a few widely used models. To begin we explain how image registration works generally in addition to describing how we measure the similarity between images. Next we outline the two different types of image registration, namely parametric and non-parametric registration, before giving some examples of each. Finally we describe the two different approaches to solving the associated minimisation problems which arise from image registration.

3.1 Introduction

Image registration is one of the most powerful tools in image processing and plays a key role in many real world applications spanning areas such as remote sensing [24,41,56,81, 120,148] and astronomy [59,69,97,121,123]. However, one area where image registration is exceptionally important is medical imaging [2,8,20,21,29,30,36,44,50,55,60–65,75, 76,94,103,104,107,111,119,124,125,132,145,146,151,152,156]. The process of image registration works by trying to find correspondences between the features in pairs, or sequences, of images. The aim of a registration model is to find the transformation which deforms one image to the other, with the goal of all images becoming similar to a single reference image. Once this transformation has been found, it can be used for other medical tasks such as anatomic image segmentation [23,53,70,105], 4D dose accumulation [1,42,58,93,114,115,134,137,154] and lung ventilation imaging [83,135].

3.1.1 The image registration model

The goal of image registration is to find the geometric transformation

$$\boldsymbol{\varphi} \equiv \boldsymbol{\varphi}(\boldsymbol{x})\colon \mathbb{R}^d \to \mathbb{R}^d$$

between the 'fixed' image R (called the reference image), and the 'moving' image T (called the template image). The aim is then for the deformed template image $T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big)$ to become similar to the reference image $R(\boldsymbol{x})$, in other words we seek

$$T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) \approx R(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \mathbb{R}^d.$$

This task is achieved by minimising some energy functional $E\big(\boldsymbol{\varphi}\big)$, with respect to the transformation $\boldsymbol{\varphi}$, consisting of a distance (or similarity) measure $\mathscr{D}\big(R, T, \boldsymbol{\varphi}\big)$ and regularisation term $\mathscr{R}\big(\boldsymbol{\varphi}\big)$. In other words, we are looking to solve the minimisation problem of the form

$$\min_{\boldsymbol{\varphi}} \big\{E\big(\boldsymbol{\varphi}\big) = \mathscr{D}\big(R, T, \boldsymbol{\varphi}\big) + \alpha\mathscr{R}\big(\boldsymbol{\varphi}\big)\big\} \tag{3.1}$$

where $\alpha \in \mathbb{R}^+$ is a weighting parameter between the two terms. Here we remark the inclusion of the regularisation term $\mathscr{R}\big(\boldsymbol{\varphi}\big)$ is necessary to ensure the minimisation problem (3.1) is well-posed in the sense of Hadamard (see §2.3).

3.1.2 Variational formulation of the registration problem

An alternative way of thinking about the registration problem (3.1), is to suppose we are trying to find the transformation of the form

$$\boldsymbol{\varphi} \equiv \boldsymbol{\varphi}\big(\boldsymbol{u}(\boldsymbol{x})\big) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \mathbb{R}^d$$

where

$$\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}) = \big[u_1(\boldsymbol{x}), \ldots, u_d(\boldsymbol{x})\big]^T \in \mathbb{R}^d$$

denotes the displacement field, and where the deformed template image is now written in the following way

$$T\big(\boldsymbol{\varphi}\big(\boldsymbol{u}(\boldsymbol{x})\big)\big) = T\big(\boldsymbol{x} + \boldsymbol{u}\big) \equiv T_{\boldsymbol{u}}. \tag{3.2}$$

Thus the minimisation problem (3.1) becomes equivalent to

$$\min_{\boldsymbol{u}} \big\{E\big(\boldsymbol{u}\big) = \mathscr{D}\big(R, T, \boldsymbol{u}\big) + \alpha\mathscr{R}\big(\boldsymbol{u}\big)\big\} \tag{3.3}$$

and we see the problem of finding the transformation $\boldsymbol{\varphi}$ becomes a task of finding the displacement field $\boldsymbol{u}$. Note once we have found the displacement field $\boldsymbol{u}$, we must use

an interpolation step to compute the deformed template image $T_{\boldsymbol{u}}$ and obtain intensity values at non-grid locations. The displacement field $\boldsymbol{u}$ is searched over the set of admissible functions $\mathcal{U}$ which minimise the functional $E\big(\boldsymbol{u}\big)$. Typically we assume the set $\mathcal{U}$ is given by a Hilbert space $\mathcal{H}$ equipped with the following inner product

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\mathcal{H}} = \int_{\Omega} \boldsymbol{u}(\boldsymbol{x}) \boldsymbol{v}(\boldsymbol{x}) \, d\Omega = \int_{\Omega} \langle \boldsymbol{u}(\boldsymbol{x}), \boldsymbol{v}(\boldsymbol{x}) \rangle_{\mathbb{R}^d} \, d\Omega$$

where $\langle \cdot, \cdot \rangle_{\mathbb{R}^d}$ denotes the Euclidean inner product. From §2.2.1, we know a necessary condition for a minimiser $\boldsymbol{u}$ of the functional $E\big(\boldsymbol{u}\big)$ is the Gâteaux derivative $\delta E(\boldsymbol{u}; \boldsymbol{\phi})$ must be zero for all variation directions $\boldsymbol{\phi} \in \mathcal{H}$. In other words we require

$$\delta E(\boldsymbol{u}; \boldsymbol{\phi}) = \lim_{\varepsilon \to 0} \left\{ \frac{E(\boldsymbol{u} + \varepsilon \boldsymbol{\phi}) - E(\boldsymbol{u})}{\varepsilon} \right\} = 0 \tag{3.4}$$

which is equivalent to

$$\nabla_{\boldsymbol{u}} E(\boldsymbol{u}) = 0$$

where $\nabla_{\boldsymbol{u}} E(\boldsymbol{u})$ defines the gradient of the functional $E(\boldsymbol{u})$. These equations are known as the Euler-Lagrange (EL) equations. Generally, we assume the energy functional $E(\boldsymbol{u})$ is of the following form

$$E(\boldsymbol{u}) = \int_{\Omega} \boldsymbol{g}\big(\boldsymbol{x}, \boldsymbol{u}(\boldsymbol{x}), \nabla \boldsymbol{u}(\boldsymbol{x})\big) \, d\Omega \tag{3.5}$$

where we also assume the function g possesses continuous partial derivatives with respect to each argument. Computing the limit (3.4), with functional of the form (3.5), along with using Green's first identity from §2.2.4 can be shown to lead to the following EL equations

$$-\nabla \cdot \nabla_{\nabla \boldsymbol{u}} \boldsymbol{g} + \nabla_{\boldsymbol{u}} \boldsymbol{g} = 0 \tag{3.6}$$

where

$$\nabla_{\boldsymbol{u}} \boldsymbol{g} \equiv \left(\frac{\partial \boldsymbol{g}}{\partial u_1}, \ldots, \frac{\partial \boldsymbol{g}}{\partial u_d} \right)^T, \; \nabla \cdot \nabla_{\nabla \boldsymbol{u}} \boldsymbol{g} \equiv \begin{bmatrix} \frac{\partial^2 \boldsymbol{g}}{\partial u_{1,1}} & \cdots & \frac{\partial^2 \boldsymbol{g}}{\partial u_{1,d}} \\ \vdots & & \vdots \\ \frac{\partial^2 \boldsymbol{g}}{\partial u_{d,1}} & \cdots & \frac{\partial^2 \boldsymbol{g}}{\partial u_{d,d}} \end{bmatrix}$$

also where

$$u_{i,j} \equiv \frac{\partial u_i}{\partial x_j}.$$

In addition to the PDE (3.6), we also get the Neumann boundary conditions

$$\big\langle \big(\nabla_{\boldsymbol{u}} \boldsymbol{g}\big), \boldsymbol{n} \big\rangle_{\mathbb{R}^d} = \big(\nabla_{\boldsymbol{u}} \boldsymbol{g}\big) \cdot \boldsymbol{n} = 0 \tag{3.7}$$

where $\boldsymbol{n}$ denotes the outward unit normal on the boundary $\partial\Omega$. Then the PDEs (3.6), along with their corresponding boundary conditions (3.7), are known as the variational formulation of the registration problem.

3.2 Similarity measures

In §3.1.1, we explained one of the two terms which make up the energy functional E is the so-called similarity measure $\mathscr{D}$ (as can be seen in (3.3)). Now the choice of this similarity measure is dependent mainly on whether we are considering the case of mono-modal or multi-modal images, in addition to whether we wish to match intensity values or landmarks within the images. In the mono-modal case the images are obtained using the same imaging modality (e.g. CT), while in the multi-modal case different imaging modalities are used to obtain the images (e.g. CT + MRI). This means in the mono-modal case image intensities are comparable, while for the multi-modal case intensities differ between the images even for the same features. As a result, we need different similarity measures for each of these two cases. Moreover, if we are using intensity values to match the images we look to match the values of every pixel in one image to their corresponding location in the other image. If we are matching landmarks however, we look to match a finite number of distinct features which appear in both images. Again we need different similarity measures for these two cases.

3.2.1 Sum of squared distances (SSD)

For the case of mono-modal images, where image intensity values are comparable, the similarity measure is given by the sum of squared distances (or SSD) [47–52, 67, 68, 72, 73, 78, 86, 90, 100, 126, 156]. The SSD measure is defined by the following

$$\mathscr{D}^{SSD}\big(R, T, \boldsymbol{u}\big) = \frac{1}{2}\int_{\Omega} \big|T_{\boldsymbol{u}} - R\big|^2 \, d\Omega \tag{3.8}$$

where $T_{\boldsymbol{u}}$ is defined in (3.2) and $|\cdot|$ denotes the Euclidean norm. It can be shown the Gâteaux derivative of (3.8) is given by

$$\nabla_{\boldsymbol{u}}\mathscr{D}^{SSD}\big(R, T, \boldsymbol{u}\big) = \nabla_{\boldsymbol{u}} T_{\boldsymbol{u}}\big(T_{\boldsymbol{u}} - R\big).$$

3.2.2 Mutual information (MI)

Now in the multi-modal image case, intensity values between the images are not comparable, and so the SSD (3.8) cannot be used as a similarity measure. Instead we can use an alternate similarity measure known as mutual information (or MI). First let us

denote

$$I_1 \equiv R(\boldsymbol{x}),\ I_2 \equiv T(\boldsymbol{x}+\boldsymbol{u}).$$

In addition let us also suppose the intensity values I_1, I_2 are continuous random variables with probability density functions (PDFs) denoted by

$$P^R(I_1),\ P^{T_{\boldsymbol{u}}}(I_2)$$

respectively. Further let us also denote the joint PDF by

$$P^{R,T_{\boldsymbol{u}}}(I_1, I_2).$$

Then the MI measure is defined to be the Kullback-Leibler distance [87,88] between the joint PDF $P^{R,T_{\boldsymbol{u}}}(I_1, I_2)$ and the product between the PDFs $P^R(I_1)$, $P^{T_{\boldsymbol{u}}}(I_2)$, i.e.

$$P^R(I_1) \cdot P^{T_{\boldsymbol{u}}}(I_2).$$

Then, the MI measure is given by the following

$$\mathscr{D}^{MI}\big(R, T, \boldsymbol{u}\big) = \int_{\mathbb{R}^2} P^{R,T_{\boldsymbol{u}}}(I_1, I_2) \log\left(\frac{P^{R,T_{\boldsymbol{u}}}(I_1, I_2)}{P^R(I_1) \cdot P^{T_{\boldsymbol{u}}}(I_2)}\right) dI_1 dI_2. \tag{3.9}$$

Here we remark if

$$P^{R,T_{\boldsymbol{u}}}(I_1, I_2) = P^R(I_1) \cdot P^{T_{\boldsymbol{u}}}(I_2) \implies \mathscr{D}^{MI}\big(R, T, \boldsymbol{u}\big) = 0$$

and we infer nothing about the random variable I_2 from the random variable I_1. Therefore, we must either seek the maximum of (3.9), or equivalently the minimum of

$$\min_{\boldsymbol{u}} \left\{ \mathscr{D}^{MI}\big(R, T, \boldsymbol{u}\big) \right\}.$$

3.2.3 Normalised cross correlation (NCC)

Suppose now, in the mono-modal image case, we are interested in matching landmarks between the images rather than intensity values. Then instead of using the SSD measure (3.8) as the distance measure, we instead use a measure called normalised cross correlation (or NCC). In NCC we begin by assuming the intensity values of the images R and T possess a linear relationship, this means they satisfy the following

$$\lambda R = \mu T_{\boldsymbol{u}}$$

where λ, $\mu \in \mathbb{R}$ are scalars. Then the so-called NCC is defined in the following way

$$\mathscr{D}^{NCC}\big(R, T, \boldsymbol{u}\big) = \frac{\langle T_{\boldsymbol{u}}, R\rangle}{|T_{\boldsymbol{u}}|\,|R|}$$

where

$$\langle T_{\boldsymbol{u}}, R\rangle \equiv \int_{\Omega} T_{\boldsymbol{u}} R \, d\Omega$$

and

$$|T_{\boldsymbol{u}}| \equiv \sqrt{\langle T_{\boldsymbol{u}}, T_{\boldsymbol{u}}\rangle},\ |R| \equiv \sqrt{\langle R, R\rangle}$$

3.3 Parametric image registration

Along with mono-modal and multi-modal image registration which depend on the type of images we are trying to register, there are another two classes of registration which depend on how regularisation is imposed. The first class is parametric image registration where the transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ is governed by a small number of parameters, and the second class is non-parametric image registration where a regularisation term $\mathscr{R}\big(\boldsymbol{\varphi}(\boldsymbol{x})\big)$ is added to the energy functional as shown in §3.1.1. In this section we briefly discuss the former class of registration, namely parametric registration, and review the rigid body and affine models. While parametric models cannot deal with non-uniform deformations well, they are useful in images involving bones since they tend to deform rigidly [85,95, 106, 130].

3.3.1 Rigid body transformations

Rigid body transformations are the simplest type of transformation which we can consider since they only allow for rotations and translations. This means we can express the transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ in the following way

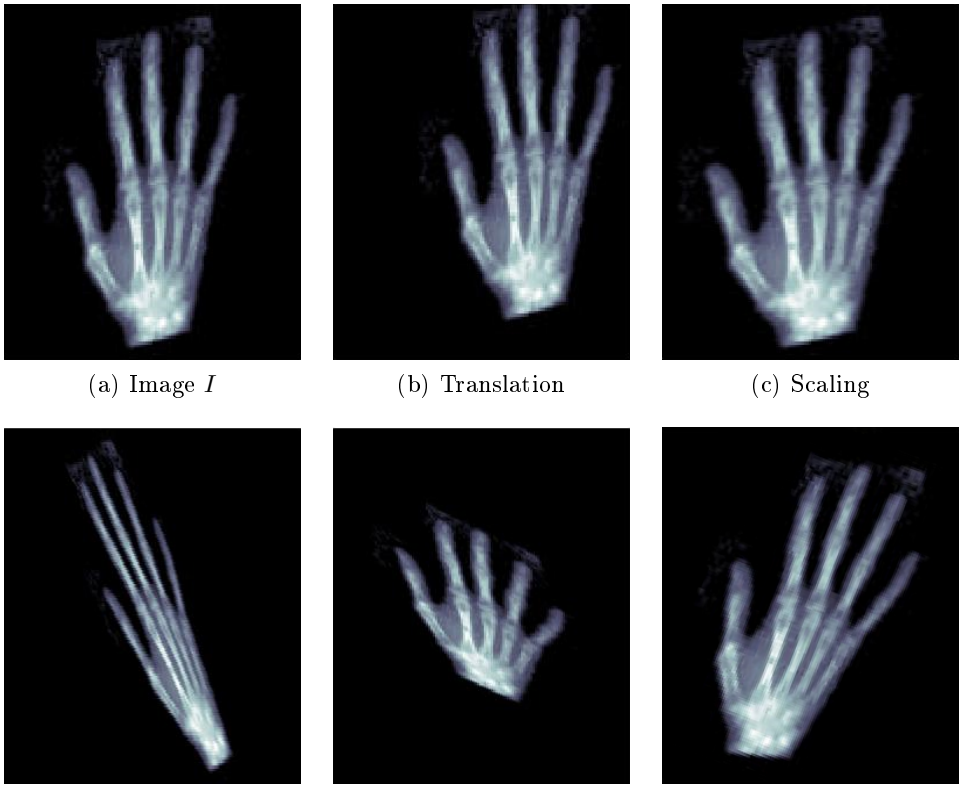
$$\boldsymbol{\varphi}(\boldsymbol{x}) = \begin{bmatrix} \varphi_1(\boldsymbol{x}) \\ \varphi_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \equiv \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$$

where θ denotes the angle of rotation and $\boldsymbol{b}$ the translation vector.

3.3.2 Affine transformations

Affine transformations can be thought of as an extension to rigid body transformations. In addition to rotations and translations, affine transformations also include scaling and shearing. Then the transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ can be expressed in the following way

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \begin{bmatrix} \varphi_1(\boldsymbol{x}) \\ \varphi_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \equiv \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$$



(a) Image I (b) Translation (c) Scaling

(d) Horizontal shearing (e) Vertical shearing (f) Rotation

Figure 3.1: Illustrations of a translation, scaling, horizontal shear, vertical shear and rotation of the image I.

where the matrix $\boldsymbol{A}$ can be written in the following form

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} S_1 & 0 \\ 0 & S_2 \end{bmatrix} \begin{bmatrix} 1 & S_3 \\ S_4 & 1 \end{bmatrix}$$

and where θ again denotes the angle of rotation, S_1, S_2 are the scaling parameters and S_3, S_4 are the shearing parameters. Again $\boldsymbol{b}$ denotes the translation vector. For examples of different types of affine transformations, see Figure 3.1.

3.4 Non-parametric image registration

Now we discuss the second class of image registration models, namely the non-parametric registration models. While parametric models are useful when the images to be registered deform rigidly, non-parametric models excel when the deformation are non-uniform such as in the case of lung images [8, 22, 38, 39, 62, 64, 91, 102, 107, 119, 132, 145, 152]. As we have already mentioned, the non-parametric registration model takes on the form shown in (3.3), with similarity measure $\mathscr{D}\big(R, T, \boldsymbol{u}\big)$ and regularisation term $\mathscr{R}(\boldsymbol{u})$. While there is a limited choice for the similarity measure (largely depending

on the images and features to be registered), for the regularisation term however the choice is not so straightforward. In the literature there is a vast amount of work on different regularisers to use such as elastic regularisers [4,11,54,100], diffusion regularisers [27,30,33,46] and optical flow [12–17,36,43,75,79,86,99,105,108,133,147,155]. Since throughout this thesis we only consider mono-modal image registration, we always take the similarity measure to be the SSD measure shown in (3.8). For the remainder of this section we briefly describe some of the most common regularisers and registration models.

3.4.1 Linear elastic image registration

The linear elastic regulariser, based upon the linearised elastic potential of the displacement field $\boldsymbol{u}$, is the most common choice of regulariser [4,11,54,100] owing to the physical properties of the model. The linear elastic regulariser is given by the following

$$\mathscr{R}^{LE}(\boldsymbol{u}) = \int_{\Omega} \frac{\mu}{4} \sum_{s,t=1}^{2} \left(\partial_{x_s} u_t + \partial_{x_t} u_s\right)^2 + \frac{\lambda}{2}\left(\nabla \cdot \boldsymbol{u}\right)^2 d\Omega \tag{3.10}$$

where μ, λ are the so-called Lamé constants with μ denoting the shear modulus and λ the bulk modulus. It can be shown the EL equations for the linear elastic model are given by the following

$$-\alpha\Big[\mu\Delta\boldsymbol{u} + \big(\mu+\lambda\big)\nabla\big(\nabla\cdot\boldsymbol{u}\big)\Big] + \boldsymbol{F}(\boldsymbol{u}) = 0$$

with the boundary conditions

$$\big(\nabla u_m + \partial_{x_m}\boldsymbol{u}\big)\cdot\boldsymbol{n} = 0 \text{ for } m = 1,2.$$

Since this model is linear, only small deformations can be found, and furthermore affine linear transformations are penalised. For more details on the linear elastic model see [4,11,100], and for a non-linear elastic model which allows for large deformations see [94,149,150].

Remark 3.4.1. *Here we remark the exact form of the force term $\boldsymbol{F}(\boldsymbol{u})$ is unknown as it is dependent upon the choose of similarity measure used, and is independent of the regularisation term chosen.*

3.4.2 Hyper-elastic image registration

The hyper-elastic regulariser, proposed by Burger et al. in [18], is a very powerful regulariser which enforces the deformation $\boldsymbol{u}$ to be diffeomorphic (i.e. a one-to-one

mapping). This regulariser is given by the following

$$\mathscr{R}^{HE}(\boldsymbol{u}) = \int_{\Omega} \alpha_1 length(\boldsymbol{u}) + \alpha_2 surface(\boldsymbol{u}) + \alpha_3 volume(\boldsymbol{u})\, d\Omega$$

where α_1, α_2, $\alpha_3 \in \mathbb{R}^+$ are weighting parameters and

$$\begin{aligned} length(\boldsymbol{u}) &= |\nabla \boldsymbol{u} - I|_F^2\,; \\ surface(\boldsymbol{u}) &= \left(\max\left\{ |\nabla \boldsymbol{u}|_F^2 - 3, 0 \right\}^2 - 3 \right)^2; \\ volume(\boldsymbol{u}) &= \left(\frac{\left(\det\left(\nabla \boldsymbol{\varphi}\right) - 1 \right)^2}{\det\left(\nabla \boldsymbol{\varphi}\right)} \right)^2 \end{aligned}$$

also where

$$\boldsymbol{\varphi} = \boldsymbol{x} + \boldsymbol{u}$$

and $|\cdot|_F$ denotes the Frobenius norm for matrices and det denotes the determinant of a matrix.

3.4.3 Diffusion image registration

The diffusion regulariser, first introduced by Fischer-Modersitzki in [47], is the simplest choice of regulariser, in addition to being very widely used [13, 15, 16, 27, 30, 33, 43, 46, 47, 105]. The diffusion regulariser is based upon the L_2-norm of the gradient of the deformation $\boldsymbol{u}$, and is given by the following

$$\mathscr{R}^{Diff}(\boldsymbol{u}) = \frac{1}{2} \int_{\Omega} \sum_{s=1}^{2} |\nabla u_s|^2 \, d\Omega \tag{3.11}$$

and yields the following EL equations

$$-\alpha \Delta \boldsymbol{u} + \boldsymbol{F}(\boldsymbol{u}) = 0$$

with Neumann boundary conditions

$$\nabla u_m \cdot \boldsymbol{n} = 0$$

for $m = 1, 2$. We remark the diffusion regulariser (3.11) can be thought of as a special case of the linear elastic regulariser (3.10) with $\mu = 1$, $\lambda = -1$. Moreover, the diffusion regulariser (3.11) also coincides with the Horn-Schunck optical flow [79] formulation, which we describe in §3.4.6. This feature is very useful when we require sequences of images to be registered (rather than simply a pair of images), which is very common in problems involving lung CT images since each image set is comprised of individual images at various phases of the breathing cycle.

3.4.4 Fischer-Modersitzki linear curvature

While all regularisers introduced in this section thus far are all first order regularisers (i.e. only dependent on the first order directional derivatives of the displacement field), the linear curvature regulariser [48, 49, 100] first proposed by Fischer-Modersitzki in [48] is second order (i.e. it depends on the second order directional derivatives). The benefit which second order regularisers have over first order regularisers is no pre-affine registration step is required (as first order regularisers penalise rigid deformations), however the trade off is second order regularisers lead to higher order PDEs which can be difficult to solve. The Fischer-Modersitzki linear curvature regulariser, which is an approximation of the mean curvature of a surface, is given by the following

$$\mathscr{R}^{LC}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega}\sum_{s=1}^{2}\left(\Delta u_s\right)^2 d\Omega. \tag{3.12}$$

Remark 3.4.2. *The mean curvature of a surface is defined by*

$$\kappa\big(u_m\big) = \nabla\cdot\frac{\nabla u_m}{\sqrt{|\nabla u_m|^2+1}} \tag{3.13}$$

for $m = 1, 2$. Supposing we have

$$|\nabla u_m| \approx 0$$

then the mean curvature (3.13) *reduces to*

$$\kappa\big(u_m\big) = \Delta u_m$$

which we see is none other than the equation corresponding to the linear curvature.

The linear curvature regulariser (3.12) leads to the following EL equations

$$\alpha\Delta^2\boldsymbol{u} + \boldsymbol{F}(\boldsymbol{u}) = 0 \tag{3.14}$$

with boundary conditions

$$\Delta u_m = 0,\ \nabla u_m\cdot\boldsymbol{n} = 0$$

for $m = 1, 2$ and where Δ^2 denotes the biharmonic operator [112]. Note the EL equations (3.14) are fourth order PDEs compared to the second order PDEs associated with the first order regularisers. There are also variations of the linear curvature regulariser (3.12), such as the Henn-Witsch curvature (see [72,74]) and mean curvature (see [34,35]), however we do not discuss the details here.

3.4.5 Total variation image registration

Another common choice of regulariser is the so-called total variation (or TV) regulariser [51, 52, 108, 116]. The TV regulariser is similar to the diffusion regulariser (3.11), with the exception of the L_2-norm being replaced with the L_1-norm. In other words we have

$$\mathscr{R}^{TV}(\boldsymbol{u}) = \frac{1}{2}\int_\Omega \sum_{s=1}^{2} |\nabla u_s|_\beta \, d\Omega = \int_\Omega \sum_{s=1}^{2} \sqrt{u_{s_{x_1}}^2 + u_{s_{x_2}}^2 + \beta} \, d\Omega \tag{3.15}$$

which results in the following EL equations

$$-\alpha \nabla \cdot \frac{\nabla \boldsymbol{u}}{|\nabla \boldsymbol{u}|_\beta} + \boldsymbol{F}(\boldsymbol{u}) = 0$$

with Neumann boundary conditions

$$\nabla u_m \cdot \boldsymbol{n} = 0$$

for $m = 1, 2$, and where $\beta \in \mathbb{R}^+$ is a small positive quantity to avoid division by zero. While the previously mentioned regularisers $\mathscr{R}^{LE}$, $\mathscr{R}^{Hyper}$, $\mathscr{R}^{Diff}$, $\mathscr{R}^{LC}$ yield smooth deformations ([33, 47–50, 72, 74, 86, 100]), they perform poorly however if discontinuities or steep gradients of $\boldsymbol{u}$ are present (for example if there are occlusions). In these instances, the TV regulariser (3.15) helps to preserve piecewise constant smoothness rather than global smoothness of $\boldsymbol{u}$, and therefore outperforms the regularisers which yield smooth deformations.

3.4.6 Optical flow

Optical flow (or optic flow) is the apparent motion of objects within a visual scene. In image registration, optical flow refers to determining the displacement fields

$$\boldsymbol{u}_1(\boldsymbol{x}), \ldots, \boldsymbol{u}_{n-1}(\boldsymbol{x})$$

of a sequence of images

$$I_1, \ldots, I_n$$

for $n \in \mathbb{N}$. Let t denote a particular frame in the image sequence, and also let $t + \Delta t$ denote the next frame in the sequence. Then, in 2D, we use the so-called brightness constancy assumption given by the following

$$I\big(x_1, x_2, t\big) = I\big(x_1 + u_1, x_2 + u_2, t + \Delta t\big) \tag{3.16}$$

where u_m denotes the displacement in the x_m direction for $m = 1, 2$ respectively. In addition we also assume the displacement field $\boldsymbol{u}$ is small, this then allows the use of a

first order Taylor expansion of (3.16) to give

$$\frac{\partial I}{\partial x_1}u_1 + \frac{\partial I}{\partial x_2}u_2 + \frac{\partial I}{\partial t}\Delta t = 0 \tag{3.17}$$

where $I \equiv I\big(x_1, x_2, t\big)$. Alternatively (3.17) can be written as

$$\frac{\partial I}{\partial x_1}V_{x_1} + \frac{\partial I}{\partial x_2}V_{x_2} + \frac{\partial I}{\partial t} = 0 \tag{3.18}$$

where V_{x_1}, V_{x_2} are the x_1, x_2 components of the velocity (or optical flow) respectively. The equation (3.18) can also be written in the following way

$$I_{x_1}V_{x_1} + I_{x_2}V_{x_2} = -I_t \text{ or } \nabla \boldsymbol{I}^T \cdot \boldsymbol{V} = -I_t. \tag{3.19}$$

where

$$I_{x_m} \equiv \frac{\partial I}{\partial x_m},\ I_t \equiv \frac{\partial I}{\partial t},\ \nabla \boldsymbol{I}^T \equiv \big[I_{x_1}, I_{x_2}\big]^T$$

for $m = 1, 2$. The equation (3.19) is known as the optical flow equation.

Remark 3.4.3. *Here we remark* (3.19) *has the same ill-posedness problem seen in* (2.13)*, and so an additional term must be added to overcome this problem.*

The two most famous methods of solving the optical flow equation (3.19) are the Horn-Schunck and Lucas-Kanade methods, which we now briefly describe.

Horn-Schunck method. In [79], Horn-Schunck proposed to formulate the optical flow equation (3.19) in the form of an energy functional (with a diffusion regulariser to overcome the ill-posedness), and sought to minimise this functional with respect to the velocity $\boldsymbol{V}$. In other words they proposed the following minimisation problem

$$\min_{\boldsymbol{V}} \left\{ E\big(\boldsymbol{V}\big) = \int_\Omega \big(I_{x_1}v_1 + I_{x_2}v_2 + I_t\big)^2 + \alpha^2 \sum_{s=1}^{2} |\nabla v_s|^2 \, d\Omega \right\} \tag{3.20}$$

where $\alpha \in \mathbb{R}^+$ is a weighting parameter. The EL equations of (3.20) are then given by

$$I_{x_m}\big(I_{x_1}v_1 + I_{x_2}v_2 + I_t\big) - \alpha^2 \Delta v_m = 0,\ m = 1, 2$$

and are solved according to the following

$$\begin{cases} \big(I_{x_1}^2 + \alpha^2\big)v_1 + I_{x_1}I_{x_2}v_2 & = \alpha^2 \bar{v}_1 - I_{x_1}I_t, \\ I_{x_1}I_{x_2}v_1 + \big(I_{x_2}^2 + \alpha^2\big)v_2 & = \alpha^2 \bar{v}_2 - I_{x_2}I_t \end{cases}$$

where we have used the following approximation for the Laplace operator

$$\Delta v_m = \bar{v}_m - v_m$$

and where $\bar{v}_m$ is a weighted average of v_m in a neighbourhood around the pixel (x_1, x_2)

for $m = 1, 2$. This then resulted in the following iterative update scheme

$$\begin{cases} v_1^{(l+1)} = \bar{v}_1^{(l)} - \dfrac{I_{x_1}\big(I_{x_1}\bar{v}_1^{(l)} + I_{x_2}\bar{v}_2^{(l)} + I_t\big)}{\alpha^2 + I_{x_1}^2 + I_{x_2}^2}, \\ \\ v_2^{(l+1)} = \bar{v}_2^{(l)} - \dfrac{I_{x_2}\big(I_{x_1}\bar{v}_1^{(l)} + I_{x_2}\bar{v}_2^{(l)} + I_t\big)}{\alpha^2 + I_{x_1}^2 + I_{x_2}^2}. \end{cases}$$

A big advantage of the Horn-Schunck method is it always produces dense flow fields, however the Horn-Schunck method is sensitive to noise. Despite this, the Horn-Schunck method is very popular in most modern optical flow models [12–16, 36, 43, 99, 105, 108, 133, 147, 155].

Lucas-Kanade method. An alternative method to the Horn-Schunck method for solving the optical flow equation (3.19), is the Lucas-Kanade method [96]. The Lucas-Kanade method uses the assumption the displacements between two image frames are small and constant within a neighbourhood of the considered point p. Then the optical flow equation (3.19) is assumed to hold for all pixels within a window centred at p, or in other words the velocity

$$\boldsymbol{V} = [v_1, v_2]^T$$

must satisfy the following

$$\begin{cases} I_{x_1}(q_1)v_1 + I_{x_2}(q_1)v_2 &= -I_t(q_1), \\ &\vdots \\ I_{x_1}(q_n)v_1 + I_{x_2}(q_n)v_2 &= -I_t(q_n) \end{cases} \tag{3.21}$$

where

$$q_1, \ldots, q_n$$

denote pixels within the window and

$$I_{x_m}(q_i),\ I_t(q_i)$$

are the image derivatives evaluated at q_i. The system (3.21) can be written in the form

$$\boldsymbol{AV} = \boldsymbol{b} \tag{3.22}$$

where

$$\boldsymbol{A} = \begin{bmatrix} I_{x_1}(q_1) & I_{x_2}(q_1) \\ \vdots & \vdots \\ I_{x_1}(q_n) & I_{x_2}(q_n) \end{bmatrix},\ \boldsymbol{V} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix},\ \boldsymbol{b} = \begin{bmatrix} -I_t(q_1) \\ \vdots \\ -I_t(q_n) \end{bmatrix}.$$

The system (3.22) is typically over-determined (i.e. more equations than unknowns), and solved using a least squares principle

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{V} = \boldsymbol{A}^T\boldsymbol{b} \text{ or } \boldsymbol{V} = \left[\boldsymbol{A}^T\boldsymbol{A}\right]^{-1}\boldsymbol{A}^T\boldsymbol{b}.$$

The matrix $\left[\boldsymbol{A}^T\boldsymbol{A}\right]^{-1}$ is referred to as the structure tensor at the point p of the image. While the Lucas-Kanade method is very robust to noise, it does not ensure the production of dense flow fields like the Horn-Schunck method does. Additionally it is not as common in modern optical flow methods when compared with the Horn-Schunck method.

3.5 General solution schemes

To solve minimisation problems such as the one in (3.3) there are two different approaches, these are the optimise-discretise and discretise-optimise approaches. For the former approach we optimise the problem first by deriving the EL equations, and then solve the discrete form of these equations on the discrete domain using some chosen method. While for the latter approach the problem is first discretised onto the discrete domain and then optimised using an optimisation scheme such as gradient descent.

3.5.1 The optimise-discretise approach

In the optimise-discretise approach the main goal is to solve the EL equations

$$\alpha\boldsymbol{\mathcal{A}}[\boldsymbol{u}] + \boldsymbol{F}(\boldsymbol{u}) = 0 \tag{3.23}$$

subject to corresponding boundary conditions. In (3.23), the force term $\boldsymbol{F}(\boldsymbol{u})$ is obtained from the Gâteaux derivative of the similarity measure $\mathscr{D}\big(R, T, \boldsymbol{u}\big)$, while $\boldsymbol{\mathcal{A}}[\boldsymbol{u}]$ is the partial differential operator obtained from the Gâteaux derivative of the regularisation term $\mathscr{R}(\boldsymbol{u})$ and $\alpha \in \mathbb{R}^+$ is the weighting parameter. There are two recognised methods to solve (3.23), namely parabolic and elliptic methods.

Parabolic method. An example of a parabolic method is the so-called time marching method [47–49, 72, 74, 90, 100] which works by introducing an artificial time variable t and computing the steady state solution of the following

$$\frac{\boldsymbol{u}\big(t^{(l+1)}\big) - \boldsymbol{u}\big(t^{(l)}\big)}{\tau} = \alpha\boldsymbol{\mathcal{A}}\big[\boldsymbol{u}\big(t^{(l+1)}\big)\big] + \boldsymbol{F}\big(\boldsymbol{u}\big(t^{(l)}\big)\big) \tag{3.24}$$

where $\tau \in \mathbb{R}^+$ denotes the time step and for convergence is required to satisfy

$$\tau < \mathcal{O}\left(\left(\frac{1}{h}\right)^2\right)$$

with h denoting the interval width in the discretisation.

Remark 3.5.1. *Here we have stated the convergence criteria for a second order PDE as this is the most common case, if however the PDE is of order four then the convergence condition would be*

$$\tau < \mathcal{O}\left(\left(\frac{1}{h}\right)^4\right).$$

A faster and more efficient scheme is the so-called additive operator splitting (or AOS) method [100, 139]. This is obtained by replacing (3.24) with

$$\boldsymbol{u}^{(l+1)} = \frac{1}{2}\sum_{s=1}^{2}\left[\boldsymbol{I} - 2\tau\alpha\boldsymbol{\mathcal{A}}_s\right]^{-1}\left[\boldsymbol{u}^{(l)} - \tau\boldsymbol{F}\big(\boldsymbol{u}^{(l)}\big)\right]$$

where $\boldsymbol{\mathcal{A}}_s$ denotes the coefficient matrix in the x_s direction respectively.

Elliptic method. An example of an elliptic method is the so-called fixed point (FP) iteration scheme [32–34, 52, 73, 156] of (3.23), which we can define by the following

$$\alpha\boldsymbol{\mathcal{L}}\big[\boldsymbol{u}^{(l+1)}\big] + \boldsymbol{F}\big(\boldsymbol{u}^{(l)}\big) = 0 \tag{3.25}$$

where $\boldsymbol{\mathcal{L}}$ denotes the linearised version of $\mathcal{A}$ at $\boldsymbol{u}^{(l)}$, and $\boldsymbol{F}\big(\boldsymbol{u}^{(l)}\big)$ denotes the linearised force term at $\boldsymbol{u}^{(l)}$ if they are non-linear.

Remark 3.5.2. *In order for the fixed point scheme* (3.25) *to converge to a solution, we require the displacement* $\boldsymbol{u}$ *be continuous.*

3.5.2 The discretise-optimise approach

Consider the discrete minimisation problem

$$\min_{\boldsymbol{u}^h}\left\{E^h(\boldsymbol{u}^h) = \mathscr{D}^h\big(R, T, \boldsymbol{u}^h\big) + \alpha\mathscr{R}^h\big(\boldsymbol{u}^h\big)\right\}.$$

Then we linearise the discrete functional $E^h(\boldsymbol{u}^h)$, about the current approximation $\boldsymbol{u}_h^{(l)}$, using the following Taylor expansion

$$E^h\big(\boldsymbol{u}_h^{(l)} + \varepsilon\boldsymbol{u}_h^{(l)}\big) = E^h\big(\boldsymbol{u}_h^{(l)}\big) + \boldsymbol{J}\big(\boldsymbol{u}_h^{(l)}\big)\varepsilon\boldsymbol{u}_h^{(l)} + \frac{1}{2}\big(\varepsilon\boldsymbol{u}_h^{(l)}\big)^T\boldsymbol{H}\big(\boldsymbol{u}_h^{(l)}\big)\varepsilon\boldsymbol{u}_h^{(l)}$$

where $\boldsymbol{J}$, $\boldsymbol{H}$ denote the Jacobian and Hessian matrices respectively. Then we update $\boldsymbol{u}_h^{(l+1)}$ according to

$$\boldsymbol{u}_h^{(l+1)} = \boldsymbol{u}_h^{(l)} + \alpha^{(l)}\varepsilon\boldsymbol{u}_h^{(l)}$$

where $\alpha^{(l)}$ is a search direction to guarantee the reduction of E^h. For Newton type methods we compute the perturbation $\varepsilon \boldsymbol{u}^{(l)}$ by solving the following normal equation

$$\boldsymbol{H}\big(\boldsymbol{u}_h^{(l)}\big)\varepsilon \boldsymbol{u}_h^{(l)} = -\boldsymbol{J}\big(\boldsymbol{u}_h^{(l)}\big).$$

Chapter 4

A more robust multigrid for diffusion type registration models

Image registration is the process of aligning pairs, or sequences, of similar images. This alignment is achieved by fixing one image (called the reference image), and then applying geometric transformations on the remaining images (called the template images) such that the template images become similar to the reference image. This technique is a very powerful tool in many real world applications spanning diverse areas such as computer imaging, weather satellite imaging [41] and especially medical imaging [8, 20, 21, 30, 61, 62] which is of interest to us. However, image registration is also one of the most difficult tasks of image processing with many challenges to be overcome. Generally image registration models can be classified into two main categories; parametric and non-parametric models. In parametric models, the transformations are global and can be described by matching a finite number of features in the images, leading to so called landmark based registration [82, 94], or the transformations are governed by a small number of parameters such as in the case of affine image registration [6, 31] (with six parameters in 2D and twelve parameters in 3D). However, the focus of this chapter will be on the latter category, namely non-parametric models.

4.1 Introduction

Denote respectively a reference and a template image (both given as grey-scale images)

$$R,\, T \in \Omega \subset \mathbb{R}^d.$$

Then the aim of image registration is to transform this T to R such that they become similar to one another, or in other words we look to find the transformation

$$\boldsymbol{\varphi}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^d$$

which satisfies the following

$$T \circ \boldsymbol{\varphi}(\boldsymbol{x}) = T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) \approx R(\boldsymbol{x}) \text{ for } \boldsymbol{x} = \big[x_1, \ldots, x_d\big]^T \in \Omega \subset \mathbb{R}^d.$$

In variational image registration the transformation $\boldsymbol{\varphi}(\boldsymbol{x})$ is equivalent to finding the displacement of every pixel $\boldsymbol{x}$ in the template image T to their corresponding pixel in the reference image R, and so we can define $\boldsymbol{\varphi}(\boldsymbol{x})$ in the following way

$$\boldsymbol{\varphi} \equiv \boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$$

where

$$\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}) = \big[u_1(\boldsymbol{x}), \ldots, u_d(\boldsymbol{x})\big]^T$$

denotes the displacement field. Then the problem of determining $\boldsymbol{\varphi}$ is equivalent to finding $\boldsymbol{u}$. From this point onward we consider only the 2D case (i.e. $d = 2$), however all ideas presented are readily extendible to the 3D case (i.e. $d = 3$). Furthermore we assume the image domain Ω is given by the unit square, i.e.

$$\Omega = [0, 1]^2 \subset \mathbb{R}^2.$$

In order to determine $\boldsymbol{u}$, the variational minimisation problem will take the following form

$$\min_{\boldsymbol{u}} \{E(\boldsymbol{u}) = \mathscr{D}(R, T, \boldsymbol{u}) + \alpha \mathscr{R}(\boldsymbol{u})\} \tag{4.1}$$

where in the energy functional $\mathscr{D}(R, T, \boldsymbol{u})$ is a distance measure, $\mathscr{R}(\boldsymbol{u})$ is the regularisation term and $\alpha \in \mathbb{R}^+$ is a weighting parameter. Note inclusion of the regularisation term is a necessity as without it the minimisation would be ill-posed in the sense of Hadamard. For simplicity we consider only mono-modal images, in other words images taken using the same imaging modality (e.g. CT), resulting in the image intensities being comparable. In the mono-modal case, the typical choice of similarity measure is the sum of squared distances (SSD) measure given by

$$\mathscr{D}(R, T, \boldsymbol{u}) = \frac{1}{2} \int_\Omega \big|T_{\boldsymbol{u}} - R\big|^2 d\Omega \ . \tag{4.2}$$

where

$$T_{\boldsymbol{u}} \equiv T(\boldsymbol{x} + \boldsymbol{u}),\ R \equiv R(\boldsymbol{x})$$

and $|\cdot|$ denotes the Euclidean norm. Here SSD is only one of many choices of similarity measure [100]. Moreover, the choice of regularisation term is less straightforward as there is a large selection to choose from [4,11,34,35,51,52,54,100,108,116] and no one is yet the best. However we select only one regularisation term, namely the diffusion regulariser, and focus on optimal solution. As for numerical implementation, the com-

mon approach is to use an optimise-discretise approach, and indeed this is the approach which we adopt.

Solutions of variational models can be computationally intensive, but such non-parametric models are worth the effort as they can produce very accurate results and are able to deal with local deformations effectively. This high computational expense is due to the need of determining the displacement of every pixel in the image. Multigrid techniques as known fast solvers have been used in previous works [51,52,68,72,73,78,86,126,156] to greatly reduce the computational cost and produce more accurate results, however few of these directly deal with the non-linearity resulting from the similarity measure (4.2). The reason for this is, while multigrid techniques and theories have been established for linear problems for a long time, achieving optimal convergence in a non-linear multigrid framework is never automatic and still poses a great challenge. However, the work done by Chumchob and Chen [33] introduced a robust multigrid framework for diffusion type variational models which treats the non-linearity directly. We propose to improve the convergence problems of the NMG method from [33] through a more in-depth and accurate analysis of the multigrid framework in addition to using an alternate coarsest solver to obtain a more efficient solution, thus resulting in a better method. Next we address how to overcome mesh folding by incorporating an additional constraint into the diffusion model presented in [33], this idea can be thought of as a simplification of the hyper-elastic model introduced in the work by Burger et al. [18]. The addition of this constraint imposes the transformation produced is regular and diffeomorphic (i.e. there is no folding). The production of diffeomorphic transformations lead to more physically meaningful results, which is particularly useful in medical imaging where folding does not occur. In this chapter, we consider one specific (yet widely used) model, namely the diffusion model to focus on our main aims:

(i) Improving the convergence of the NMG method from [33];

(ii) Development of a fast NMG method for a refined diffusion model which controls folding.

As we have already mentioned, there are many other choices for the regularisation term $\mathscr{R}(\boldsymbol{u})$ [4,11,34,35,51,52,54,100,108,116], each offering a different model and with their own distinct benefits and drawbacks. In particular, we mention

Total variation (TV) [51,52,108,116].

$$\mathscr{R}^{TV}(\boldsymbol{u}) = \sum_{s=1}^{2} \int_{\Omega} \left|\nabla u_s\right| d\Omega$$

where $|\cdot|$ denotes the Euclidean norm;

Linear elastic (LE) [4,11,54,100].

$$\mathscr{R}^{LE}(\boldsymbol{u}) = \int_\Omega \frac{\mu}{4} \sum_{s,t=1}^{2} \left(\partial_{x_s} u_t + \partial_{x_t} u_s\right)^2 + \frac{\lambda}{2}\left(\nabla \cdot \boldsymbol{u}\right)^2 d\Omega$$

where μ, λ are Lamé constants;

Mean curvature (MC) [34,35].

$$\mathscr{R}^{MC}(\boldsymbol{u}) = \frac{1}{2} \int_\Omega \sum_{s=1}^{2} \nabla \cdot \left(\frac{\nabla u_s}{\sqrt{\left|\nabla u_s\right|^2 + \beta}} \right)^2 d\Omega$$

where β is some small positive quantity.

While each such model might be solved by a NMG framework, achieving optimal efficiency would require further work and development.

The remainder of this chapter will be set out in the following way. In §4.2 we introduce the formulation of the registration model focusing specifically on the diffusion model. Next in §4.3 we discuss the non-linear multigrid (NMG) framework applied to the diffusion model, along with a detailed analysis to highlight how we can improve the convergence of the Chumchob-Chen NMG method. Then in §4.4 we formulate our non-folding constraint model, and also present an optimisation for the implementation of the constraint. §4.5 will comprise of tests and comparisons with our proposed work, and finally in §4.6 we present a summary of this chapter.

4.2 Review of the registration model and algorithm of [33]

The diffusion regulariser is a popular choice among variational models as seen in the works [12,15–17,79]. It imposes a simple smoothness constraint upon the displacement field and is given by the following

$$\mathscr{R}^{Diff}(\boldsymbol{u}) = \frac{1}{2} \int_\Omega \sum_{s=1}^{2} \left|\nabla u_s\right|^2 d\Omega \ . \tag{4.3}$$

In fact, the diffusion model is one of the few models which coincides with models from optical flow frameworks (see [15, 16, 79] as examples), this is particularly useful when the registration of image sequences are required. The diffusion model is given by the following minimisation problem

$$\min_{\boldsymbol{u}} \left\{ E^{Diff}(\boldsymbol{u}) = \mathscr{D}(R, T, \boldsymbol{u}) + \alpha \mathscr{R}^{Diff}(\boldsymbol{u}) = \frac{1}{2} \int_\Omega \left|T_{\boldsymbol{u}} - R\right|^2 + \alpha \sum_{s=1}^{2} \left|\nabla u_s\right|^2 d\Omega \right\} \tag{4.4}$$

where

$$T_{\boldsymbol{u}} \equiv T(\boldsymbol{x}+\boldsymbol{u}),\ R \equiv R(\boldsymbol{x}).$$

The corresponding Euler-Lagrange (EL) equations are derived from the following limits

$$\begin{cases} \lim\limits_{\varepsilon_1 \to 0} \dfrac{E^{Diff}(u_1+\varepsilon_1\phi_1, u_2) - E^{Diff}(u_1,u_2)}{\varepsilon_1} = 0, \\ \lim\limits_{\varepsilon_2 \to 0} \dfrac{E^{Diff}(u_1, u_2+\varepsilon_2\phi_2) - E^{Diff}(u_1,u_2)}{\varepsilon_2} = 0 \end{cases}$$

which can be shown to result in the following integrals

$$\int_\Omega \phi_m \Big[\partial_{u_m} T_{\boldsymbol{u}} \big[T_{\boldsymbol{u}} - R\big] - \alpha \Delta u_m\Big]\, d\Omega + \alpha \int_{\partial\Omega} \phi_m \big(\nabla u_m \cdot \boldsymbol{n}\big)\, dS = 0. \tag{4.5}$$

for $m = 1, 2$. Then, we can apply the fundamental lemma of calculus of variations to (4.5) to obtain the EL equations

$$-\alpha \Delta u_m + F_m(\boldsymbol{u}) = 0 \tag{4.6}$$

with Neumann boundary conditions

$$\nabla u_m \cdot \boldsymbol{n} = 0$$

where $\boldsymbol{n}$ denotes the outward unit normal and

$$F_m(\boldsymbol{u}) = \partial_{u_m} T_{\boldsymbol{u}} \big[T_{\boldsymbol{u}} - R\big] \tag{4.7}$$

denote the force terms, for $m = 1, 2$.

4.2.1 Optimise-discretise approach for the diffusion model

We consider a numerical approximation to the EL equations (4.6) by discretising the image domain Ω^h into a uniform $n \times n$ mesh with interval width h, using a finite difference (FD) method. The size of the mesh is chosen to be equal to the dimension of the image (e.g. 512×512 to coincide with resolution of the given images) and in general need not be square, however we only consider the case of square images as this is common for medical image slices. Using the following central FD approximations

$$\big(\partial^h_{u_1} T^h_{\boldsymbol{u}}\big)_{i,j} \approx \frac{1}{2h}\Big[\big(T^h_{\boldsymbol{u}}\big)_{i+1,j} - \big(T^h_{\boldsymbol{u}}\big)_{i-1,j}\Big],\ \big(\partial_{u_2} T^h_{\boldsymbol{u}}\big)_{i,j} \approx \frac{1}{2h}\Big[\big(T^h_{\boldsymbol{u}}\big)_{i,j+1} - \big(T^h_{\boldsymbol{u}}\big)_{i,j-1}\Big],$$

$$\big(\Delta^h u^h_m\big)_{i,j} \approx \frac{1}{h^2}\Big[\big(u^h_m\big)_{i,j-1} + \big(u^h_m\big)_{i-1,j} - 4\big(u^h_m\big)_{i,j} + \big(u^h_m\big)_{i+1,j} + \big(u^h_m\big)_{i,j+1}\Big] \tag{4.8}$$

at a general discrete point (i, j), leads to the following discrete versions of the EL equations (4.6)

$$-\alpha\big(\Delta^h u_m^h\big)_{i,j} + \big(F_m(\boldsymbol{u}^h)\big)_{i,j} = 0 \tag{4.9}$$

with the discrete force terms

$$\big(F_m(\boldsymbol{u}^h)\big)_{i,j} = \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_{i,j}\left[\big(T_{\boldsymbol{u}}^h\big)_{i,j} - \big(R^h\big)_{i,j}\right] \tag{4.10}$$

for $m = 1, 2$ and $i,\, j = 2, \ldots, n-1$.

4.2.2 The collective pointwise smoother

The term smoother, which stems from multigrid theory, is nothing but an iterative solver. In [33] the lexicographic Gauss-Seidel (GSLEX) method was employed to solve the linear part of the system (4.9) through an inner iteration loop, and a fixed point iteration scheme to solve the non-linear part through an outer iteration loop. In a lexicographical ordering system, a general discrete point (i, j) as in (4.10) is linked to the global index

$$k = (j-2)(n-1) + (i-1)$$

with n the size of the discrete image dimensions. An illustration of the lexicographical ordering system can be seen in Figure 4.1. Using a lexicographical ordering on the discrete system (4.9), results in the following

$$-\alpha\big(\Delta^h u_m^h\big)_k + \big(F_m(\boldsymbol{u}^h)\big)_k = 0 \tag{4.11}$$

for $m = 1, 2$. Now to solve the non-linear part of this system, we employ the following semi-implicit fixed point iteration scheme

$$-\alpha\big(\Delta^h u_m^h\big)_k^{(l+1)} + \big(F_m(\boldsymbol{u}^h)\big)_k^{(l+1)} = 0$$

where

$$\big(F_m(\boldsymbol{u}^h)\big)_k^{(l+1)} = \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\left[\big(T_{\boldsymbol{u}}^h\big)_k^{(l+1)} - \big(R^h\big)_k\right] \tag{4.12}$$

with

$$\big(T_{\boldsymbol{u}}^h\big)_k^{(l+1)} \equiv \big(T^h(\boldsymbol{x} + \boldsymbol{u}^{(l+1)})\big)_k.$$

The key question addressed in [33] was how to treat the non-linear term $\left(T^h_{\boldsymbol{u}}\right)^{(l+1)}_k$ in a GSLEX scheme. It proposed to use the first order approximations:

$$\left(T^h_{\boldsymbol{u}}\right)^{(l+1)}_k \approx \left(T^h_{\boldsymbol{u}}\right)^{(l)}_k + \sum_{s=1}^{2} \left(\partial^h_{u_s} T^h_{\boldsymbol{u}}\right)^{(l)}_k \left[\left(u^h_s\right)^{(l+1)}_k - \left(u^h_s\right)^{(l)}_k\right]$$

which we substitute back into the discrete force terms (4.10) leading to the following discrete system

$$-\alpha\left(\Delta^h u^h_m\right)^{(l+1)}_k + \left(\partial^h_{u_m} T^h_{\boldsymbol{u}}\right)^{(l)}_k \Bigg[\left(T^h_{\boldsymbol{u}}\right)^{(l)}_k + \sum_{s=1}^{2} \left(\partial^h_{u_s} T^h_{\boldsymbol{u}}\right)^{(l)}_k \left[\left(u^h_s\right)^{(l+1)}_k - \left(u^h_s\right)^{(l)}_k\right] - \left(R^h\right)_k\Bigg] = 0 \quad (4.13)$$

with

$$\left(T^h_{\boldsymbol{u}}\right)^{(l)}_k \equiv \left(T^h(\boldsymbol{x} + \boldsymbol{u}^{(l)})\right)_k$$

etc. for $m = 1, 2$. Then to compute the $(l+1)$ updates in (4.13), we use a GSLEX based method. Unfortunately, such an iterative method is not effective as a standalone solver since solving the discrete system of PDEs (4.11) pixel-wise can lead to a very high computational cost, especially for large images. This fact is well-known for simpler PDEs such as the Poisson equation (corresponding to $F_m = 0$ and $h \to 0$). One natural way of reducing the cost of calculating the displacement field is a NMG method in which this (slow) iterative method is used as a smoother.

There has already been a lot of work regarding the implementation of NMG methods [52, 68, 72, 73, 86] for related models, each having its own uni-grid iterative solver. However most of these works do not address the non-linearity in the similarity measure directly, instead linear diagonal terms or augmented systems are used. Chumchob and Chen [33] proposed a robust solver which does directly deal with this non-linearity arising from the SSD term, however an inaccurate analysis of the NMG method led to a less than optimal convergence rate for the NMG method which we demonstrate in the next section.

4.2.3 The NMG method

There are two theoretical principles driving multigrid methods for linear PDEs. The first is, although standard iterative methods such as the Jacobi and GS methods have poor convergence rates when used as independent solvers, they are effective at smoothing out any high frequency error components within a small number of iterations. This property leads to the second key principle of multigrid methods, namely low frequency error components can be well approximated on a coarser grid. Naturally an approximate and accurate solution on a coarser grid can then be interpolated back to the fine grid

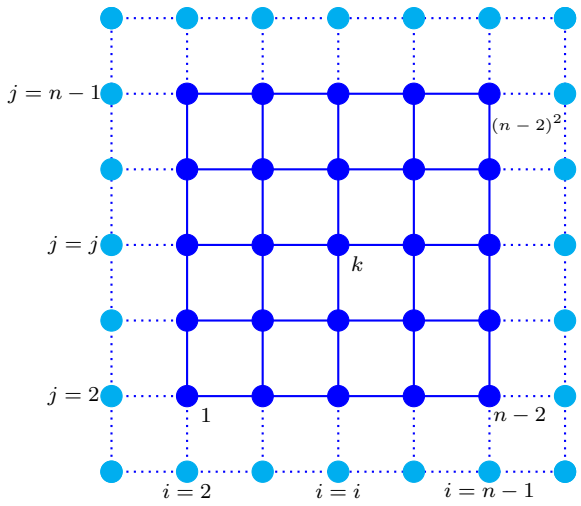


Figure 4.1: Illustration of how the domain Ω^h is discretised by $n \times n$ grid points. The dashed light blue line represents the boundary $\partial\Omega^h$ of the discrete domain, with the light blue circle points representing the used boundary points, and the solid dark blue lines show the $(n-2) \times (n-2)$ grid corresponding to the interior points represented by the solid dark blue circles. The indexing on the interior points show how the global index k is ordered lexicographically.

to approximate the original problem; this two-grid approach is significantly cheaper than working solely on the fine grid. In fact this strategy allows us to obtain a more accurate approximation efficiently as we can perform a larger number of iterations on the coarser grid in less time when compared with iterating on the fine grid alone. This fine-coarse-fine strategy, known as the two-grid V-cycle (see [10] for details), can be repeated on the coarse grid to interact with even coarser grids until some coarsest grid with few points.

While multigrid frameworks are known, and indeed very easy to implement for linear cases, problems like (4.6) which are highly non-linear prove significantly more difficult to develop a converging NMG method. Now we present the FAS-NMG algorithm of [33] for (4.11) before we highlight the omissions in the analysis which resulted in an over-estimated smoothing rate (thus leading to a less optimal NMG method with slower convergence rate), and include our more accurate analysis to overcome this problem. Here FAS stands for 'full approximation scheme' by A. Brandt for solving a non-linear operator equation.

Remark 4.2.1. *The FAS can be thought of as a generalisation of the linear multigrid schemes discussed in §2.6. In fact if the operator being considered is linear, then the FAS directly reduces to the linear two-grid correction scheme.*

First consider a two grid setting where Ω^h denotes a fine grid and Ω^H a coarse grid with

$$\boldsymbol{h} = (h_1, h_2) = \left(\frac{1}{n-1}, \frac{1}{n-1}\right), \ \boldsymbol{H} = 2\boldsymbol{h}.$$

Also denote the system (4.11) using the following operator notation on Ω^h

$$\boldsymbol{\mathcal{N}}^h \boldsymbol{u}^h = \boldsymbol{\mathcal{G}}^h$$

where

$$\boldsymbol{\mathcal{N}}^h = \begin{bmatrix} \big(\mathcal{N}_1^h\big)_k \\ \big(\mathcal{N}_2^h\big)_k \end{bmatrix}, \boldsymbol{u}^h = \begin{bmatrix} \big(u_1^h\big)_k \\ \big(u_2^h\big)_k \end{bmatrix}, \boldsymbol{\mathcal{G}}^h = \begin{bmatrix} \big(g_1^h\big)_k \\ \big(g_2^h\big)_k \end{bmatrix}$$

and with

$$\begin{aligned} (\mathcal{N}_1^h)_k = \big(F_1(\boldsymbol{u}^h)\big)_k - \alpha\big(\Delta^h u_1^h\big)_k,\ (\mathcal{N}_2^h)_k = \big(F_2(\boldsymbol{u}^h)\big)_k - \alpha\big(\Delta^h u_2^h\big)_k, \\ (g_1^h)_k = (g_2^h)_k = 0 \end{aligned}$$

for $k = 1, 2, \ldots, (n-2)^2$. The main steps of the FAS-NMG are as followed:

Smoothing step. Apply the iterative method (4.13) on grid Ω^h starting from some initial guess. This is the pre-smoothing step required to obtain a smooth approximation

$$\tilde{\boldsymbol{u}}^h = [\tilde{u}_1^h, \tilde{u}_2^h]^T$$

which has non-linear residual

$$\boldsymbol{r}^h = \boldsymbol{\mathcal{G}}^h - \boldsymbol{\mathcal{N}}^h(\tilde{\boldsymbol{u}}^h).$$

To improve this smooth approximation, it remains to compute the algebraic error (or the residual correction)

$$\boldsymbol{e}^h \equiv [e_1^h, e_2^h]^T = \boldsymbol{u}^h - \tilde{\boldsymbol{u}}^h$$

which cannot be computed directly on the fine grid Ω^h.

Restriction. Since only smooth errors can be well approximated on a coarser grid, we first solve the FAS coarse grid residual equation

$$\boldsymbol{\mathcal{N}}^H[\boldsymbol{u}^H] \equiv \boldsymbol{\mathcal{N}}^H[\tilde{\boldsymbol{u}}^H + \boldsymbol{e}^H] = \boldsymbol{r}^H + \boldsymbol{\mathcal{N}}^H[\tilde{\boldsymbol{u}}^H] \equiv \boldsymbol{\mathcal{G}}^H \tag{4.14}$$

where

$$\tilde{\boldsymbol{u}}^H = \mathcal{R}_h^H \tilde{\boldsymbol{u}}^h,\ \boldsymbol{r}^H = \mathcal{R}_h^H \boldsymbol{r}^h$$

and $\mathcal{R}_h^H$ denotes the restriction operator, which we take to be the full-weighted restriction operator, defined by the following stencil

$$\mathcal{R}_h^H = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_h^H. \tag{4.15}$$

Coarse grid solution. For a two-grid method (or in a multigrid setting where Ω^H is the coarsest level and computations are inexpensive), the above coarse grid equation (4.14) must be solved accurately to obtain the solution $\boldsymbol{u}^H$. Based on this $\boldsymbol{u}^H$, and its initial guess $\tilde{\boldsymbol{u}}^H$, we obtain the residual correction via the relation

$$\boldsymbol{e}^H = \boldsymbol{u}^H - \tilde{\boldsymbol{u}}^H. \tag{4.16}$$

Interpolation. Now we wish to use (4.16) to correct the approximations on the fine grid Ω^h; we do this by interpolating the corrections using bilinear interpolation. In other words we compute

$$\boldsymbol{e}^h = \mathcal{I}_H^h \boldsymbol{e}^H, \mathcal{I}_H^h = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}_H^h.$$

Once the corrections have been interpolated to the next fine grid level, we use them to update the current grid level approximations via

$$\boldsymbol{u}^h = \tilde{\boldsymbol{u}}^h + \boldsymbol{e}^h.$$

After the approximations have been corrected, we use a post-smoothing step to remove any interpolation errors. This process of interpolation, correction and smoothing is repeated until the approximations on the original grid level have been corrected and smoothed, thus resulting in our final solution $\boldsymbol{u}^h$.

Remark 4.2.2. *According to the work done in [71], there are three conditions which need to be satisfied regarding the orders of the restriction and interpolation methods for a convergent NMG. For a PDE of order M, we require*

(i) $m_R + m_I \geq M$; (ii) $m_I \geq M$ and $m_R \geq 0$; (iii) $m_R \geq M$ and $m_I \geq 0$

where m_R, m_I denote the high frequency orders of the restriction and interpolation schemes respectively. In our case we have $m_R = m_I = 2$, for the full-weighted restriction and bilinear interpolation operators respectively, and so all three conditions are satisfied.

A summary of the FAS-NMG algorithm, for the case of an arbitrary number of levels, can be seen in Algorithm 6.

In [33], the coarsest solver adopted was an additive operator splitting (AOS) method [100, 139]. For the diffusion model, the AOS methods takes the following form

$$u_m^{(l+1)} = \frac{1}{2}\sum_{s=1}^{2} [I - 2\tau\alpha L_{x_s}]^{-1} \left[u_m^{(l)} - \tau F_m(\boldsymbol{u}^{(l)}) + \tau g_m\right]$$

where I denotes the identity operator, $\tau > 0$ the time-step which is determined using a forward difference approximation of the time derivative $\partial_t u_m$, g_m the RHS coming from

Algorithm 6 $\boldsymbol{u}_{level}^{(l+1)} \leftarrow FASNMG(level, \mu, \boldsymbol{u}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 steps

$$\tilde{\boldsymbol{u}}_{level}^{(l)} \leftarrow Smooth(\boldsymbol{u}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1)$$

2: Coarse grid correction

Compute the residual $\boldsymbol{r}_{level}^{(l)} = \boldsymbol{F}_{level} - \boldsymbol{\mathcal{N}}_{level}(\tilde{\boldsymbol{u}}_{level}^{(l)})$

Restrict residual and smooth approximations $\boldsymbol{r}_{level-1}^{(l)} = \mathcal{R}_{level}^{level-1}\boldsymbol{r}_{level}^{(l)}$, $\tilde{\boldsymbol{u}}_{level-1}^{(l)} = \mathcal{R}_{level}^{level-1}\tilde{\boldsymbol{u}}_{level}^{(l)}$

Set $level \rightarrow level - 1$

Compute RHS of coarse grid PDE $\boldsymbol{f}_{level-1} = \boldsymbol{r}_{level-1}^{(l)} + \boldsymbol{\mathcal{N}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(l)}$

Compute an approximation $\bar{\boldsymbol{u}}_{level-1}^{(l)}$ to the coarse grid PDE $\boldsymbol{\mathcal{N}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(l)} = \boldsymbol{r}_{level-1}^{(l)}$

Solve residual equation on coarse grid to obtain approximations $\bar{\boldsymbol{u}}_{level-1}^{(l)}$

3: **if** $level = 1$ **then**

Use a direct or fast iterative solver to obtain the high accuracy solutions $\boldsymbol{u}_{level-1}^{(l)}$

4: **else** $level > 1$ Repeat the FAS-cycle procedure recursively to the next level using $\tilde{\boldsymbol{u}}_{level-1}^{(l)}$ as an initial approximation i.e.

$$\bar{\boldsymbol{u}}_{level}^{(l)} \leftarrow FAScycle(level - 1, \mu, \tilde{\boldsymbol{u}}_{level-1}^{(l)}, \boldsymbol{\mathcal{N}}_{level-1}, \boldsymbol{F}_{level-1}, \nu_1, \nu_2)$$

5: **end if**

Compute the correction $\boldsymbol{e}_{level-1}^{(l)} = \boldsymbol{u}_{level-1}^{(l)} - \tilde{\boldsymbol{u}}_{level-1}^{(l)}$

Interpolate the correction to next fine grid level $\boldsymbol{e}_{level}^{(l)} = \mathcal{I}_{level-1}^{level}\boldsymbol{e}_{level-1}^{(l)}$

Update current grid level approximations using correction $\hat{\boldsymbol{u}}_{level}^{(l)} = \tilde{\boldsymbol{u}}_{level}^{(l)} + \boldsymbol{e}_{level}^{(l)}$

6: Post-smoothing step by performing ν_2 steps

$$\boldsymbol{u}_{level}^{(l+1)} \leftarrow Smooth(\hat{\boldsymbol{u}}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_2)$$

the NMG framework, $F_m(\boldsymbol{u})$ the force terms given in (4.7) for $m = 1, 2$ and

$$L_{x_s} = \partial_{x_s x_s}$$

denote the parts of the discrete Laplace operator in the x_s direction for $s = 1, 2$ respectively. The above equations are updated along the x_1, x_2 directions separately, thus leading to the system

$$\begin{cases} \left[I - 2\tau\alpha L_{x_1}\right] u_{m,p_1}^{\left(k+\frac{1}{2}\right)} = \dfrac{1}{2}\left[u_m^{(l)} - \tau F_m^{(l)}(\boldsymbol{u}) + \tau g_m\right], \\ \\ \left[I - 2\tau\alpha L_{x_2}\right] u_{m,p_2}^{\left(k+\frac{1}{2}\right)} = \dfrac{1}{2}\left[u_m^{(l)} - \tau F_m^{(l)}(\boldsymbol{u}) + \tau g_m\right] \end{cases} \tag{4.17}$$

with the updates

$$u_m^{(l+1)} = \frac{1}{2}\left(u_{m,p_1}^{(k+\frac{1}{2})} + u_{m,p_2}^{(k+\frac{1}{2})}\right)$$

for $m = 1, 2$.

H-ellipticity for proposed smoother. The computation of the h-ellipticity for a given smoother scheme is a very important step in determining whether the smoother scheme is suitable for use as a pointwise error smoothing scheme within the multigrid

framework. We now perform this calculation for the proposed smoother described in §4.2.2.

Let us begin by writing the linearised system of PDEs (4.13) in the following operator form

$$\boldsymbol{\mathcal{L}}_h^{Diff}\boldsymbol{u}^h = \boldsymbol{\mathcal{G}}_h^{Diff} \tag{4.18}$$

where

$$\begin{aligned}\boldsymbol{\mathcal{L}}_h^{Diff} &= \begin{bmatrix} -\alpha\Delta^h + \sigma_{11}^h & \sigma_{12}^h \\ \sigma_{12}^h & -\alpha\Delta^h + \sigma_{22}^h \end{bmatrix}, \boldsymbol{u}^h = \begin{bmatrix} u_1^h \\ u_2^h \end{bmatrix}, \\ \boldsymbol{\mathcal{G}}_h^{Diff} &= \begin{bmatrix} g_1^h - F_1^{Diff}(\boldsymbol{u}^h) \\ g_2^h - F_2^{Diff}(\boldsymbol{u}^h) \end{bmatrix}\end{aligned} \tag{4.19}$$

with

$$\begin{aligned}F_m^{Diff}(\boldsymbol{u}^h) &= (\partial_{u_m}^h T_{\boldsymbol{u}}^h)^2 u_m^h - (\partial_{u_m}^h T_{\boldsymbol{u}}^h)[T_{\boldsymbol{u}}^h - R^h], \\ \sigma_{pq}^h &= (\partial_{u_m}^h T_{\boldsymbol{u}}^h)(\partial_{u_q}^h T_{\boldsymbol{u}}^h),\ g_m^h = 0\end{aligned} \tag{4.20}$$

for m, p, $q = 1, 2$. Applying the discrete linear operator $\boldsymbol{\mathcal{L}}_h^{Diff}$, to the grid functions $\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta})$ gives

$$\boldsymbol{\mathcal{L}}_h^{Diff}\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{L}}}_h^{Diff}(\boldsymbol{\theta})\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta}) \tag{4.21}$$

with Fourier symbol

$$\hat{\boldsymbol{\mathcal{L}}}_h^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h - \alpha\hat{\mathscr{L}}^h(\boldsymbol{\theta}) & \sigma_{12}^h \\ \sigma_{12}^h & \sigma_{22}^h - \alpha\hat{\mathscr{L}}^h(\boldsymbol{\theta}) \end{bmatrix} \tag{4.22}$$

where $\hat{\mathscr{L}}^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the discrete Laplace operator Δ^h. We compute the h-ellipticity from the following

$$\mathscr{E}_h^{Diff}(\boldsymbol{\mathcal{L}}_h^{Diff}) = \frac{\min\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}}{\max\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}} \tag{4.23}$$

where

$$\boldsymbol{\Theta} = [-\pi, \pi)^2,\ \boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2.$$

It can be shown

$$\det\left(\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})\right) = -\alpha^2(\hat{\mathscr{L}}^h(\boldsymbol{\theta}))^2 + \alpha c_1^h(\hat{\mathscr{L}}^h(\boldsymbol{\theta})) + c_2^h \tag{4.24}$$

where

$$c_1^h = \sigma_{11}^h + \sigma_{22}^h,\ c_2^h = \sigma_{11}^h \sigma_{22}^h. \tag{4.25}$$

Using well known results, we can show

$$\begin{aligned} -\hat{\mathscr{L}}^h(\boldsymbol{\theta}) &= \frac{2}{h^2}\big[2 - \big(\cos\theta_1 + \cos\theta_2\big)\big], \\ \min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{high}} \Big\{\big(-\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)\Big\} &= \frac{2}{h^2},\ \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \Big\{\big(-\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)\Big\} = \frac{8}{h^2}. \end{aligned} \tag{4.26}$$

Substituting (5.34) and (4.24) back into (6.43), in addition to taking the limit as $h \to 0$, we get

$$\lim_{h\to 0} \Big\{\mathscr{E}_h^{Diff}\big(\boldsymbol{\mathcal{L}}_h^{Diff}\big)\Big\} = \lim_{h\to 0} \left\{\frac{4\alpha^2 + \mathcal{O}(h)}{64\alpha^2 + \mathcal{O}(h)}\right\} = \frac{1}{16}. \tag{4.27}$$

Since the h-ellipticity value (4.27) is bounded away from 0, as $h \to 0$, and is therefore independent of the values of α, h, σ_{pq}^h for $p, q = 1, 2$. This means the results do not depend on the given images R, T, the choice of the weighting parameter α or the mesh interval h. Thus we conclude the smoother (4.13) is sufficient for use as a pointwise error smoothing procedure.

4.3 An improved analysis of the NMG algorithm of [33]

As mentioned, Algorithm 6 as implemented by Chumchob and Chen [33] could still be slow to converge to a solution from new experiments. We found a major part of this convergence problem was the result of an inaccurate analysis of the smoothing rate, which resulted in an over-estimation of the rate. Re-evaluating the analysis of the NMG method, in addition to building in some new components, led to our NMG algorithm with a vastly improved convergence rate.

In this section we outline our more detailed and accurate analysis of the NMG framework. We do this by analysing two key components of the NMG algorithm (namely the smoothing rate of the smoother and the coarsest grid solver), which leads to an optimal NMG method.

4.3.1 Smoother analysis using local Fourier analysis (LFA)

We begin our analysis of the NMG method by showing an improved, and more accurate, LFA of the smoother scheme which was described in [33]. A discrete error (e.g. residual) function on a grid can be written as a sum of two terms:

(i) High frequency error components (not visible if the problem is restricted to a coarser grid);

(ii) Low frequency error components (can be accurately represented on a coarser grid).

The sole purpose of the smoother, within a MG framework, is to remove any high frequency error components. LFA is used to measure how effective a given smoother scheme is.

Although LFA was originally designed to analyse discrete linear operator equations, it was extended by A. Brandt (see [131]) to study non-linear operators via a 'freezing' of localised coefficients. To start we first assume we are working on an infinite grid, this allows us to remove any influence from the boundary conditions. Next we assume the discrete form of a non-linear operator, with variable coefficients, can be replaced locally by an operator with constant coefficients and extended to the infinite grid. We need to ensure all high frequency error components are removed prior to restriction to a coarse grid. As a result it is imperative we know how effective our relaxation scheme is at smoothing out the errors so we can adjust the number of sweeps required for the pre- and post-smoothing steps. Using LFA we obtain a value μ which we define to be the smoothing factor for a given relaxation scheme.

LFA for pointwise smoother from [33]. While the smoother we described in §4.2.2 is the same as the one used in [33], we found the smoother analysis in [33] contained an omission which lead to a very over-optimistic smoothing rate (practically to a slow convergence if using it as a guide). In [33], the discrete system (4.11) was written in the following way

$$\boldsymbol{\mathcal{N}}_{+}^{h}\boldsymbol{u}_{new}^{h}+\boldsymbol{\mathcal{N}}_{0}^{h}\boldsymbol{u}_{new}^{h}+\boldsymbol{\mathcal{N}}_{-}^{h}\boldsymbol{u}_{old}^{h}=\boldsymbol{\mathcal{G}}^{h}$$

where $\boldsymbol{u}_{new}^{h}$, $\boldsymbol{u}_{old}^{h}$ denote the current and previous approximations of $\boldsymbol{u}^{h}$ respectively, and

$$\boldsymbol{\mathcal{N}}_{+}^{h}=\begin{bmatrix}-\alpha\mathscr{L}_{+}^{h} & 0\\ 0 & -\alpha\mathscr{L}_{+}^{h}\end{bmatrix},\ \boldsymbol{\mathcal{N}}_{0}^{h}=\begin{bmatrix}-\alpha\mathscr{L}_{0}^{h}+\sigma_{11}^{h} & \sigma_{12}^{h}\\ \sigma_{12}^{h} & -\alpha\mathscr{L}_{0}^{h}+\sigma_{22}^{h}\end{bmatrix},$$

$$\boldsymbol{\mathcal{N}}_{-}^{h}=\begin{bmatrix}-\alpha\mathscr{L}_{-}^{h} & 0\\ 0 & -\alpha\mathscr{L}_{-}^{h}\end{bmatrix},\ \boldsymbol{\mathcal{G}}^{h}=\begin{bmatrix}g_{1}^{h}-F_{1}^{h}\\ g_{2}^{h}-F_{2}^{h}\end{bmatrix} \tag{4.28}$$

with

$$\sigma_{pq}^{h}=\partial_{u_p}T_{\boldsymbol{u}}^{h}\partial_{u_q}T_{\boldsymbol{u}}^{h}$$

and g_m^h denoting the RHS coming from the NMG scheme, F_m^h the discrete force terms as given in (4.10) and where $\mathscr{L}_+^h$, $\mathscr{L}_0^h$, $\mathscr{L}_-^h$ define the following stencils

$$\mathscr{L}_+^h = \frac{1}{h^2}\begin{bmatrix}0&0&0\\1&0&0\\0&1&0\end{bmatrix}, \mathscr{L}_0^h = \frac{1}{h^2}\begin{bmatrix}0&0&0\\0&-4&0\\0&0&0\end{bmatrix}, \mathscr{L}_-^h = \frac{1}{h^2}\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}. \tag{4.29}$$

for p, q, $m = 1, 2$. The smoothing rate in [33] was then calculated on a 32×32 grid after a total of five outer and five inner iteration loops had been performed, thus resulting in an average smoothing rate of $\mu_{avg} \approx 0.5$ when taking $\alpha = \frac{1}{10}$. However, in the analysis of [33] we notice the $(u_m)_k^{(l)}$ terms, which result from the linearisation of the SSD term, where not included in the smoothing rate calculation. This omission meant the obtained rate of 0.5 was a vast over-estimation of the actual smoothing rate, and as a result led to an under-estimation of the number of pre-smoothing steps required before restriction. This means when we restrict the problem to a coarser grid, there are still high frequency error components remaining on the fine grid which have not been removed, and so the coarse grid correction we obtain is much less accurate thus leading to more NMG cycles being required to reach an accurate solution. This omission, as we now show, has a noticeable effect on the smoothing rate.

Revised LFA for pointwise smoother from §4.2.2. Here we repeat the analysis of the smoothing rate, with the $(u_m)_k^{(l)}$ terms included, in order to illustrate the impact the addition of these terms have on the smoothing rate. We begin by writing the discrete equations (4.11) in the following form

$$\boldsymbol{\mathcal{N}}^h \boldsymbol{u}^h + \boldsymbol{\mathcal{M}}^h \boldsymbol{u}^h = \boldsymbol{\mathcal{G}}^h \tag{4.30}$$

where $\boldsymbol{\mathcal{G}}^h$ is as in (4.28), and

$$\boldsymbol{\mathcal{N}}^h = \begin{bmatrix}-\alpha\Delta^h + \sigma_{11}^h & \sigma_{12}^h \\ \sigma_{12}^h & -\alpha\Delta^h + \sigma_{22}^h\end{bmatrix}, \boldsymbol{\mathcal{M}}^h = \begin{bmatrix}-\sigma_{11}^h & -\sigma_{12}^h \\ -\sigma_{12}^h & -\sigma_{22}^h\end{bmatrix}.$$

Using the following representation of the discrete Laplace operator

$$\Delta^h \equiv \mathscr{L}_+^h + \mathscr{L}_0^h + \mathscr{L}_-^h$$

with $\mathscr{L}_+^h$, $\mathscr{L}_0^h$, $\mathscr{L}_-^h$ as defined in (4.29), then we can express (4.30) in the following way

$$\boldsymbol{\mathcal{N}}_+^h \boldsymbol{u}_{new}^h + \boldsymbol{\mathcal{N}}_0^h \boldsymbol{u}_{new}^h + \boldsymbol{\mathcal{N}}_-^h \boldsymbol{u}_{old}^h + \boldsymbol{\mathcal{M}}^h \boldsymbol{u}_{old}^h = \boldsymbol{\mathcal{G}}^h \tag{4.31}$$

and subtracting (4.31) from (4.30) yields the local error equation given by

$$\big[\boldsymbol{\mathcal{N}}_+^h + \boldsymbol{\mathcal{N}}_0^h\big]\boldsymbol{e}_{new}^h = -\big[\boldsymbol{\mathcal{N}}_-^h + \boldsymbol{\mathcal{M}}^h\big]\boldsymbol{e}_{old}^h \tag{4.32}$$

where $\boldsymbol{\mathcal{N}}_+^h$, $\boldsymbol{\mathcal{N}}_0^h$, $\boldsymbol{\mathcal{N}}_-^h$ are as defined in (4.28) and

$$\boldsymbol{e}_*^h = \left[e_{1\,*}^h, e_{2\,*}^h\right]^T. \tag{4.33}$$

Using Fourier components, we can rewrite (4.32) in the following way

$$\left[\hat{\boldsymbol{\mathcal{N}}}_+^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{N}}}_0^h(\boldsymbol{\theta})\right]\psi_{\boldsymbol{\theta}}^{new}\exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j\right) = -\left[\hat{\boldsymbol{\mathcal{N}}}_-^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}^h(\boldsymbol{\theta})\right]\psi_{\boldsymbol{\theta}}^{old}\exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j\right) \tag{4.34}$$

where

$$\lambda_m = \frac{2\theta_m\pi}{h},\ \boldsymbol{i} = \sqrt{-1},\ \boldsymbol{\theta} \in \boldsymbol{\Theta} = [-\pi, \pi)^2$$

and $\psi_{\boldsymbol{\theta}}^*$ are Fourier coefficients for $m = 1, 2$. From here we determine the local smoothing rate μ_{loc} using the following

$$\mu_{\max} = \max_{\text{loc}}\mu_{loc},\ \mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\rho\big(\hat{\boldsymbol{S}}^h(\boldsymbol{\theta})\big)\big|\boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}$$

where

$$\boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$$

also with $\rho(\cdot)$ denoting the spectral radius, and the amplification matrix $\hat{\boldsymbol{S}}^h(\boldsymbol{\theta})$ given by

$$\hat{\boldsymbol{S}}^h(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{N}}}_+^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{N}}}_0^h(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{N}}}_-^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}^h(\boldsymbol{\theta})\right]$$

with

$$\hat{\boldsymbol{\mathcal{N}}}_+^h(\boldsymbol{\theta}) = \begin{bmatrix} -\frac{\alpha}{h^2}\left(e^{-\boldsymbol{i}\lambda_1} + e^{-\boldsymbol{i}\lambda_2}\right) & 0 \\ 0 & -\frac{\alpha}{h^2}\left(e^{-\boldsymbol{i}\lambda_1} + e^{-\boldsymbol{i}\lambda_2}\right) \end{bmatrix},$$

$$\hat{\boldsymbol{\mathcal{N}}}_-^h(\boldsymbol{\theta}) = \begin{bmatrix} -\frac{\alpha}{h^2}\left(e^{\boldsymbol{i}\lambda_1} + e^{\boldsymbol{i}\lambda_2}\right) & 0 \\ 0 & -\frac{\alpha}{h^2}\left(e^{\boldsymbol{i}\lambda_1} + e^{\boldsymbol{i}\lambda_2}\right) \end{bmatrix},$$

$$\hat{\boldsymbol{\mathcal{N}}}_0^h(\boldsymbol{\theta}) = \begin{bmatrix} \frac{4\alpha}{h^2} + \sigma_{11}^h & \sigma_{12}^h \\ \sigma_{12}^h & \frac{4\alpha}{h^2} + \sigma_{22}^h \end{bmatrix},\ \hat{\boldsymbol{\mathcal{M}}}^h(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma_{11}^h & -\sigma_{12}^h \\ -\sigma_{12}^h & -\sigma_{22}^h \end{bmatrix}. \tag{4.35}$$

Implementing the revised local smoothing rate formulae, under the same conditions which were used in [33], we obtained an average and maximum smoothing rate of

$$\mu_{avg} \approx 0.69854,\ \mu_{max} \approx 0.74762$$

respectively. By the smoothing rate of 0.5 in [33] within each outer iteration, five inner iterations would result in a reduction of the error by 0.0313 which appeared satisfactory. However five inner iterations would reduce the error by only 0.17 and 0.23 respectively

using our new smoothing rates μ_{avg} and μ_{max}. In order to reduce to the level of error claimed in [33], we estimate we would require up to twelve inner iterations. So we see the original analysis in [33] resulted in the estimated number of pre-smoothing steps being roughly half of the number of steps which would actually be required to reduce the error to quoted level.

4.3.2 Convergence analysis of two coarsest grid solvers by LFA

Next we give a simple solution to the challenging problem of estimating the convergence rate of a non-linear iterative method. Here we remark this analysis was not performed in [33]. Consequently, we can compare methods and guide the number of iterations to be prescribed on the coarsest grid, similar to how we use the smoothing rate to guide the number of smoothing steps required. Recall the AOS solver (4.17) was used by Chumchob and Chen in [33] as the solver on the coarsest grid. Here we propose to use a fixed point type solver on the coarsest grid instead.

Our coarsest grid solver. From §4.2.2 we have the following lexicographically ordered discrete system of linear equations

$$-\alpha\big(\Delta^H u_m^H\big)_k^{(l+1)} + \big(\partial_{u_m}^H T_{\boldsymbol{u}}^H\big)_k^{(l)}\Bigg[\big(T_{\boldsymbol{u}}^H\big)_k^{(l)} + \sum_{s=1}^{2}\big(\partial_{u_s}^H T_{\boldsymbol{u}}^H\big)_k^{(l)}\Big[\big(u_s^H\big)_k^{(l+1)} - \big(u_s^H\big)_k^{(l)}\Big] - \big(R^H\big)_k\Bigg] = 0$$

for $m = 1, 2$. After using the FD approximations (4.8), we can express these equations as matrix equations

$$\boldsymbol{A}^H\boldsymbol{u}^H = \boldsymbol{F}^H$$

where

$$\boldsymbol{u}^H,\, \boldsymbol{F}^H \in \mathbb{R}^{2(n-2)^2\times 1}$$

are block column vectors and

$$\boldsymbol{A}^H \in \mathbb{R}^{2(n-2)^2\times 2(n-2)^2}$$

is the block system matrix with the following structure

$$\boldsymbol{A}^H = \begin{bmatrix} \boldsymbol{A}_1^H & \boldsymbol{B}^H \\ \boldsymbol{B}^H & \boldsymbol{A}_2^H \end{bmatrix},\ \boldsymbol{u}^H = \begin{bmatrix} \boldsymbol{u}_1^H \\ \boldsymbol{u}_2^H \end{bmatrix},\ \boldsymbol{F}^H = \begin{bmatrix} \boldsymbol{F}_1^H \\ \boldsymbol{F}_2^H \end{bmatrix}$$

where

$$\boldsymbol{u}_m^H,\, \boldsymbol{F}_m^H \in \mathbb{R}^{(n-2)^2\times 1}$$

are column vectors and

$$\boldsymbol{A}_m^H,\, \boldsymbol{B}^H \in \mathbb{R}^{(n-2)^2\times(n-2)^2}$$

are the block tri-diagonal and diagonal matrices respectively with the following structure

$$\boldsymbol{A}_m^H = \begin{bmatrix} A_{m_2} & I_1 & & \\ I_1 & \ddots & \ddots & \\ & \ddots & \ddots & I_1 \\ & & I_1 & A_{m_{n-1}} \end{bmatrix},\ \boldsymbol{B}^H = \begin{bmatrix} B_{m_2} & & \\ & \ddots & \\ & & B_{m_{n-1}} \end{bmatrix}$$

$$\boldsymbol{u}_m^H = \begin{bmatrix} \left(u_m^H\right)_{k_2(2)} \\ \vdots \\ \left(u_m^H\right)_{k_i(j)} \\ \vdots \\ \left(u_m^H\right)_{k_{n-2}(n-2)} \end{bmatrix},\ \boldsymbol{f}_m^H = \begin{bmatrix} \left(f_m^H\right)_{k_2(2)} \\ \vdots \\ \left(f_m^H\right)_{k_i(j)} \\ \vdots \\ \left(f_m^H\right)_{k_{n-2}(n-2)} \end{bmatrix}$$

with matrices

$$A_{m_j},\, B_{m_j},\, I_1 \in \mathbb{R}^{(n-2)\times(n-2)}$$

of structure

$$A_{m_j} = \begin{bmatrix} (a_m^H)_{k_2(j)} & -\frac{\alpha}{H^2} & & \\ -\frac{\alpha}{H^2} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\alpha}{H^2} \\ & & -\frac{\alpha}{H^2} & (a_m^H)_{k_{n-1}(j)} \end{bmatrix},$$

$$B_{m_j} = \begin{bmatrix} \left(b^H\right)_{k_2(j)} & & \\ & \ddots & \\ & & \left(b^H\right)_{k_{n-1}(j)} \end{bmatrix},\ I_1 = -\frac{\alpha}{H^2}\begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

with

$$(a_m^H)_{k_i(j)} = \left[\left(\partial_{u_m}^H T_{\boldsymbol{u}}^H\right)^2\right]_{k_i(j)} + \frac{4\alpha}{H^2},\ \left(b^H\right)_{k_i(j)} = \left(\partial_{u_1}^H T_{\boldsymbol{u}}^H\right)_{k_i(j)}\left(\partial_{u_2}^H T_{\boldsymbol{u}}^H\right)_{k_i(j)}$$

and where

$$k_i(j) = (j-2)(n-1) + (i-1)$$

denotes a general lexicographically ordered discrete point (i, j), as shown in Figure 4.1. Then our proposed algorithm is as shown in Algorithm 7.

In order to demonstrate the improvement in the convergence rate of our proposed coarsest grid solver over the AOS scheme used in [33], we first need a way to measure the

Algorithm 7 $\boldsymbol{u}_H^{(l+1)} \leftarrow DirectSolve(R^H, T^H, \boldsymbol{u}_H^{(l)}, \boldsymbol{\mathcal{G}}^H, \alpha, IMAX, Tol)$

Initialise $\boldsymbol{u}_H^{(l)} = \boldsymbol{u}_H^{(l)}$
    Construct discrete Laplacian parts of sparse matrices $\boldsymbol{A}_m^H$
for $l = 1, \cdots, IMAX$ **do**
    Deform template image using $\boldsymbol{u}_H^{(l)} \rightarrow T_{\boldsymbol{u}}^H$
    Compute FD approximations for derivatives of $T_{\boldsymbol{u}}^H \rightarrow \partial_{u_1}^H T_{\boldsymbol{u}}^H$, $\partial_{u_2}^H T_{\boldsymbol{u}}^H$
    Compute RHS f_m (matrices) and then convert to column vectors $\boldsymbol{f}_m^H$
    Add remaining diagonal parts to $\boldsymbol{A}_m^H$
    Compute $\boldsymbol{u}_{m\,H}^{(l+1)} \rightarrow \boldsymbol{u}_{m\,H}^{(l+1)} = \left[\boldsymbol{A}_m^H\right]^{-1} \boldsymbol{f}_m^H$
    Reshape $\boldsymbol{u}_{m\,H}^{(l+1)}$ to matrices $u_{m\,H}^{(l+1)}$
    if $\left|\boldsymbol{u}_{1\,H}^{(l+1)} - \boldsymbol{u}_{1\,H}^{(l)}\right|_2^2 < Tol$ and $\left|\boldsymbol{u}_{2\,H}^{(l+1)} - \boldsymbol{u}_{2\,H}^{(l)}\right|_2^2 < Tol$ **then**
    Exit for loop
    end if
end for

convergence rate. To do this we employ LFA to estimate the convergence rates of both our proposed solver and the AOS solver. The purpose is to discriminate between these two estimations. Unfortunately due to the non-linearity of the problem we are unable to obtain a sharp measure of the convergence rate, and so using LFA to obtain an approximation is the best option. It should be remarked LFA used for this convergence analysis is only viable on a coarse grid (e.g. 8×8 mesh) as the rate is not sharp especially on a fine grid (e.g. 128×128 mesh).

Analysis of the proposed coarsest grid solver. To estimate the convergence rate $\mathcal{P}$ of a given solver, we follow a similar method to the one in the smoother analysis shown in §4.3.1. In other words we must evaluate the amplification matrix $\hat{\boldsymbol{S}}^H(\boldsymbol{\theta})$ at every discrete interior point (i, j) for $i,\, j = 2, \ldots, n-1$ where n denotes the size of the image dimensions. However, where we restricted $\boldsymbol{\theta}$ to only consider the high frequency range $\boldsymbol{\Theta}_{high}$ in the smoother analysis, now we consider $\boldsymbol{\theta}$ over the entire Fourier domain $\boldsymbol{\Theta}$. Since our proposed direct solver is based upon the pointwise smoother shown in §4.2.2, the derivation of the amplification matrix $\hat{\boldsymbol{S}}^H(\boldsymbol{\theta})$ is very similar to the one shown in §4.3.1. Then, the convergence rate for our proposed direct solver can be estimated locally by the following

$$\mathcal{P}_{D\,\max} = \max_{\text{loc}} \mathcal{P}_{D\,loc},\; \mathcal{P}_{D\,loc} \equiv \mathcal{P}_{D\,loc}(\boldsymbol{\theta}) = \sup\left\{\rho\big(\hat{\boldsymbol{S}}^H(\boldsymbol{\theta})\big) \big| \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}$$

where

$$\boldsymbol{\Theta} \in [-\pi, \pi)^2$$

also with $\rho(\cdot)$ denoting the spectral radius and $\hat{\boldsymbol{S}}^H(\boldsymbol{\theta})$ the amplification matrix given by

$$\hat{\boldsymbol{S}}^H(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{N}}}_+^H(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{N}}}_0^H(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{N}}}_-^H(\boldsymbol{\theta})\right]^{-1} \hat{\boldsymbol{\mathcal{M}}}^H(\boldsymbol{\theta})$$

with

$$\hat{\boldsymbol{\mathcal{N}}}_{+}^{H}(\boldsymbol{\theta}),\, \hat{\boldsymbol{\mathcal{N}}}_{0}^{H}(\boldsymbol{\theta}),\, \hat{\boldsymbol{\mathcal{N}}}_{-}^{H}(\boldsymbol{\theta}),\, \hat{\boldsymbol{\mathcal{M}}}^{H}(\boldsymbol{\theta})$$

as in (4.35) and where $\boldsymbol{H}$ denotes the interval width on the coarse grid.

Analysis of the block formulation of our proposed coarsest grid solver. Previously in order to estimate the convergence rate for the pointwise case, we would have a single equation of the form shown in (4.34) for each discrete interior point from which we would determine the amplification matrix, now however we construct the amplification matrix from a single system of equations with the following structure

$$\boldsymbol{B}^{H}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{new} = \boldsymbol{C}^{H}\boldsymbol{\Psi}_{\boldsymbol{\theta}}^{old}$$

where

$$\boldsymbol{B}^{H},\, \boldsymbol{C}^{H} \in \mathbb{R}^{2(n-2)^2 \times 2(n-2)^2},\, \boldsymbol{\Psi}_{\boldsymbol{\theta}}^{*} \in \mathbb{R}^{2(n-2)^2 \times 1}$$

are block matrices and block column vectors respectively with structure

$$\boldsymbol{B}^{H} = \begin{bmatrix} \boldsymbol{B}_{1}^{H} & \boldsymbol{D}^{H} \\ \boldsymbol{D}^{H} & \boldsymbol{B}_{2}^{H} \end{bmatrix},\, \boldsymbol{C}^{H} = \begin{bmatrix} \boldsymbol{C}_{1}^{H} & \boldsymbol{D}^{H} \\ \boldsymbol{D}^{H} & \boldsymbol{C}_{2}^{H} \end{bmatrix},\, \boldsymbol{\Psi}_{\boldsymbol{\theta}}^{*} = \begin{bmatrix} \boldsymbol{\psi}_{\boldsymbol{\theta}}^{*} \\ \boldsymbol{\psi}_{\boldsymbol{\theta}}^{*} \end{bmatrix}$$

with

$$\boldsymbol{B}_{m}^{H},\, \boldsymbol{C}_{m}^{H},\, \boldsymbol{D}^{H} \in \mathbb{R}^{(n-2)^2 \times (n-2)^2},\, \boldsymbol{\psi}_{\boldsymbol{\theta}}^{*} \in \mathbb{R}^{(n-2)^2 \times 1}$$

given by

$$\boldsymbol{B}_{m}^{H} = \begin{bmatrix} B_{m_2}^{H} & J_{1}^{H} & & \\ J_{2}^{H} & \ddots & \ddots & \\ & \ddots & \ddots & J_{1}^{H} \\ & & J_{2}^{H} & B_{m_{n-1}}^{H} \end{bmatrix},\, \boldsymbol{C}^{H} = \begin{bmatrix} C_{m_2}^{H} & & \\ & \ddots & \\ & & C_{m_{n-1}}^{H} \end{bmatrix},$$

$$\boldsymbol{D}^{H} = \begin{bmatrix} D_{2}^{H} & & \\ & \ddots & \\ & & D_{n-1}^{H} \end{bmatrix},\, \boldsymbol{\psi}_{\boldsymbol{\theta}}^{*} = \begin{bmatrix} \left(\psi_{\boldsymbol{\theta}}^{*}\right)_{1} \\ \vdots \\ \left(\psi_{\boldsymbol{\theta}}^{*}\right)_{k} \\ \vdots \\ \left(\psi_{\boldsymbol{\theta}}^{*}\right)_{(n-2)^2} \end{bmatrix}$$

and where

$$B_{m_j}^{H},\, C_{m_j}^{H},\, D_{j}^{H},\, J_{m}^{H} \in \mathbb{R}^{(n-2)\times(n-2)}$$

are given by

$$B^H_{m_j} = \begin{bmatrix} \left(b^H_m\right)_{k_2(j)} & -\frac{\alpha}{H^2}e^{\boldsymbol{i}\lambda_1} & & \\ -\frac{\alpha}{H^2}e^{-\boldsymbol{i}\lambda_1} & \ddots & \ddots & \\ & \ddots & \ddots & -\frac{\alpha}{H^2}e^{\boldsymbol{i}\lambda_1} \\ & & -\frac{\alpha}{H^2}e^{-\boldsymbol{i}\lambda_1} & \left(b^H_m\right)_{k_{n-1}(j)} \end{bmatrix},$$

$$C^H_{m_j} = \begin{bmatrix} \left(c^H_m\right)_{k_2(j)} & & \\ & \ddots & \\ & & \left(c^H_m\right)_{k_{n-1}(j)} \end{bmatrix}, \; D^H_j = \begin{bmatrix} \left(d^H\right)_{k_2(j)} & & \\ & \ddots & \\ & & \left(d^H\right)_{k_{n-1}(j)} \end{bmatrix},$$

$$J^H_1 = \begin{bmatrix} -\frac{\alpha}{H^2}e^{\boldsymbol{i}\lambda_2} & & \\ & \ddots & \\ & & -\frac{\alpha}{H^2}e^{\boldsymbol{i}\lambda_2} \end{bmatrix}, \; J^H_2 = \begin{bmatrix} -\frac{\alpha}{H^2}e^{-\boldsymbol{i}\lambda_2} & & \\ & \ddots & \\ & & -\frac{\alpha}{H^2}e^{-\boldsymbol{i}\lambda_2} \end{bmatrix}$$

with

$$\left(b^H_m\right)_{k_i(j)} = \left[\left(\partial^H_{u_m} T^H_{\boldsymbol{u}}\right)^2\right]_{k_i(j)} + \frac{4\alpha}{H^2}, \; \left(c^H_m\right)_{k_i(j)} = \left[\left(\partial^H_{u_m} T^H_{\boldsymbol{u}}\right)^2\right]_{k_i(j)},$$
$$\left(d^H\right)_{k_i(j)} = \left(\partial^H_{u_1} T^H_{\boldsymbol{u}}\right)_{k_i(j)} \left(\partial^H_{u_2} T^H_{\boldsymbol{u}}\right)_{k_i(j)}, \; \lambda_m = \frac{2\theta_m \pi}{n}$$

and

$$k_i(j) = (j-2)(n-1) + (i-1)$$

for $m = 1, 2$ and $i, j = 2, \ldots, n-1$. Then the convergence rate $\mathcal{P}_B$ for the block formulation of our direct solver is estimated from the following

$$\mathcal{P}_B \equiv \mathcal{P}_B(\boldsymbol{\theta}) = \sup\left\{\rho\big(\hat{\boldsymbol{S}}^H(\boldsymbol{\theta})\big) \big| \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}$$

with amplification matrix

$$\hat{\boldsymbol{S}}^H(\boldsymbol{\theta}) = \left[\boldsymbol{B}^H\right]^{-1} \boldsymbol{C}^H.$$

On this coarsest grid, n is small so estimating $\mathcal{P}_B$ is feasible.

Convergence analysis for AOS solver. We again remark an analysis to estimate the convergence of the coarsest solver in [33] was not performed. From [33], the AOS scheme for the diffusion model is shown in (4.17) for $m = 1, 2$. We use a similar method to the one shown in §4.3.1 to derive the amplification matrix for the AOS method. However, since the AOS scheme solves along the x_1 and x_2 directions separately, we obtain two convergence rates $\mathcal{P}_{A_1}$, $\mathcal{P}_{A_2}$ corresponding to each of these directions. We

start by expressing the discrete versions of (4.17) by the following system

$$\boldsymbol{\mathcal{N}}_m^H \boldsymbol{u}_{p_m}^H + \boldsymbol{\mathcal{M}}_m^H \boldsymbol{u}_{p_m}^H = \boldsymbol{\mathcal{G}}_m^H \tag{4.36}$$

with

$$\boldsymbol{\mathcal{N}}_m^H = \begin{bmatrix} 1 - 2\tau\alpha\partial_{x_m x_m}^H & 0 \\ 0 & 1 - 2\tau\alpha\partial_{x_m x_m}^H \end{bmatrix},$$
$$\boldsymbol{\mathcal{M}}_m^H = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \boldsymbol{\mathcal{G}}_m^H = \begin{bmatrix} \tau g_1^H - \tau F_1^H(\boldsymbol{u}) \\ \tau g_2^H - \tau F_2^H(\boldsymbol{u}) \end{bmatrix}$$

where g_m^H are the discrete RHS coming from the NMG method and $F_m^H(\boldsymbol{u})$ are the discrete force terms given in (4.10). The x_1, x_2 directions of the discrete Laplace operator can be represented by

$$\partial_{x_m x_m}^H = \mathscr{L}_{m+}^H + \mathscr{L}_{m0}^H + \mathscr{L}_{m-}^H$$

where $\mathscr{L}_{m+}^H$, $\mathscr{L}_{m0}^H$, $\mathscr{L}_{m-}^H$ define the following stencils

$$\mathscr{L}_{1+}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathscr{L}_{10}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathscr{L}_{1-}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix},$$
$$\mathscr{L}_{2+}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \mathscr{L}_{20}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \mathscr{L}_{2-}^H = \frac{1}{H^2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.37}$$

Then we can write (4.36) in the following way

$$\boldsymbol{\mathcal{N}}_{m+}^H \boldsymbol{u}_{p_m\, new}^H + \boldsymbol{\mathcal{N}}_{m0}^H \boldsymbol{u}_{p_m\, new}^H + \boldsymbol{\mathcal{N}}_{m-}^H \boldsymbol{u}_{p_m\, old}^H + \boldsymbol{\mathcal{M}}_m^H \boldsymbol{u}_{p_m\, old}^H = \boldsymbol{\mathcal{G}}_m^H \tag{4.38}$$

where $\boldsymbol{u}_{p_m\, new}^H$, $\boldsymbol{u}_{p_m\, old}^H$ denote the current and previous approximations of $\boldsymbol{u}_{p_m}^H$ in the x_m directions respectively, and

$$\boldsymbol{\mathcal{N}}_{m+}^H = \begin{bmatrix} -2\tau\alpha\mathscr{L}_{m+}^H & 0 \\ 0 & -2\tau\alpha\mathscr{L}_{m+}^H \end{bmatrix}, \boldsymbol{\mathcal{N}}_{m-}^H = \begin{bmatrix} -2\tau\alpha\mathscr{L}_{m-}^H & 0 \\ 0 & -2\tau\alpha\mathscr{L}_{m-}^H \end{bmatrix},$$
$$\boldsymbol{\mathcal{N}}_{m0}^H = \begin{bmatrix} 1 - 2\tau\alpha\mathscr{L}_{m0}^H & 0 \\ 0 & 1 - 2\tau\alpha\mathscr{L}_{m0}^H \end{bmatrix}, \boldsymbol{\mathcal{M}}_m^H = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$

for $m = 1, 2$. Using a similar process to the one shown in §4.3.1, for computing the smoothing rate, we estimate the convergence rate from the following

$$\mathcal{P}_{A\,\max} = \max_{\text{loc}} \mathcal{P}_{A\,loc},\ \mathcal{P}_{A\,loc} = \frac{1}{2}\big(\mathcal{P}_{A_1\,loc} + \mathcal{P}_{A_2\,loc}\big),$$
$$\mathcal{P}_{A_m\,loc} \equiv \mathcal{P}_{A_m\,loc}(\boldsymbol{\theta}) = \sup\Big\{\rho\big(\hat{\boldsymbol{S}}_m^H(\boldsymbol{\theta})\big) \big| \boldsymbol{\theta} \in \boldsymbol{\Theta}\Big\}$$

		AOS Solver		Direct Solver (Pointwise)		Direct Solver (Block)	
Grid Size	$\boldsymbol{\alpha}$	$\boldsymbol{\mathcal{P}_A}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$	$\boldsymbol{\mathcal{P}_D}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$	$\boldsymbol{\mathcal{P}_B}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$
4×4	$\frac{1}{10}$	0.99915	2709/5417/8124	0.40511	3/6/8	0.14573	2/3/4
	$\frac{1}{20}$	0.99957	5355/10708/16062	0.51635	4/7/11	0.26136	2/4/6
	$\frac{1}{30}$	0.99971	7940/15879/23817	0.61297	5/10/15	0.35084	3/5/7
8×8	$\frac{1}{10}$	0.99937	3655/7309/10962	0.82924	13/25/37	0.41411	3/6/8
	$\frac{1}{20}$	0.99968	7195/14390/21584	0.90661	24/47/71	0.63061	5/10/15
	$\frac{1}{30}$	0.99979	10965/21928/32892	0.93578	35/70/105	0.76812	9/18/27
16×16	$\frac{1}{10}$	0.99947	4344/8688/13031	0.97391	88/175/262	0.99636	632/1262/1894
	$\frac{1}{20}$	0.99973	8528/17055/25582	0.98679	174/647/520	0.99784	1065/2130/3195
	$\frac{1}{30}$	0.99982	12792/25583/38374	0.99116	260/519/778	0.99853	1565/3131/4696

Table 4.1: Comparison 1 of convergence rates (averaged over five FAS-NMG cycles) for the Chumchob-Chen AOS solver and our direct solver. For each solver the convergence rates and number of iterations required to reach tolerances of 10^{-1}, 10^{-2}, 10^{-3} are shown for multiple α values on various coarsest grid sizes for the lung CT example (Example 2 in Figure 4.3).

		AOS Solver		Direct Solver (Pointwise)		Direct Solver (Block)	
Grid Size	$\boldsymbol{\alpha}$	$\boldsymbol{\mathcal{P}_A}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$	$\boldsymbol{\mathcal{P}_D}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$	$\boldsymbol{\mathcal{P}_B}$	***Tol*** $\mathbf{10^{-1}/10^{-2}/10^{-3}}$
4×4	$\frac{1}{10}$	0.99915	2708/5416/8123	0.65472	6/11/17	0.32791	3/5/7
	$\frac{1}{20}$	0.99957	5355/10708/16061	0.79307	10/20/30	0.51094	4/7/11
	$\frac{1}{30}$	0.99971	7940/15879/23817	0.85177	15/29/44	0.62553	5/10/15
8×8	$\frac{1}{10}$	0.99937	3655/7309/10962	0.94157	39/77/115	0.70146	7/13/20
	$\frac{1}{20}$	0.99968	7195/14390/21584	0.96925	74/148/222	0.88868	20/40/59
	$\frac{1}{30}$	0.99979	10965/21928/32892	0.97894	109/217/325	0.97361	87/173/259
16×16	$\frac{1}{10}$	0.99947	4344/8688/13031	0.98925	214/427/640	0.99756	943/1886/2828
	$\frac{1}{20}$	0.99973	8528/17055/25582	0.99463	428/856/1283	0.99872	1798/3596/5394
	$\frac{1}{30}$	0.99982	12792/25583/38374	0.99643	644/1288/1932	0.99941	3902/7804/11705

Table 4.2: Comparison 2 of convergence rates (averaged over five FAS-NMG cycles) for the Chumchob-Chen AOS solver and our direct solver. For each solver the convergence rates and number of iterations required to reach tolerances of 10^{-1}, 10^{-2}, 10^{-3} are shown for multiple α values on various coarsest grid sizes for the hand example (Example 3 in Figure 4.3).

where $\rho(\cdot)$ again denotes the spectral radius, and $\hat{\boldsymbol{S}}_m^h(\boldsymbol{\theta})$ denote the amplification matrices given by

$$\hat{\boldsymbol{S}}_m^H(\boldsymbol{\theta}) = -\big[\hat{\boldsymbol{\mathcal{N}}}_{m+}^H(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{N}}}_{m0}^H(\boldsymbol{\theta})\big]^{-1}\big[\hat{\boldsymbol{\mathcal{N}}}_{m-}^H(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_m^H(\boldsymbol{\theta})\big]$$

with

$$\hat{\boldsymbol{\mathcal{N}}}_{m+}^H(\boldsymbol{\theta}) = \begin{bmatrix} -\frac{2\tau\alpha}{H^2}e^{-\boldsymbol{i}\lambda_m} & 0 \\ 0 & -\frac{2\tau\alpha}{H^2}e^{-\boldsymbol{i}\lambda_m} \end{bmatrix}, \hat{\boldsymbol{\mathcal{N}}}_{m0}^H(\boldsymbol{\theta}) = \begin{bmatrix} 1+\frac{4\tau\alpha}{H^2} & 0 \\ 0 & 1+\frac{4\tau\alpha}{H^2} \end{bmatrix},$$
$$\hat{\boldsymbol{\mathcal{N}}}_{m-}^H(\boldsymbol{\theta}) = \begin{bmatrix} -\frac{2\tau\alpha}{H^2}e^{\boldsymbol{i}\lambda_m} & 0 \\ 0 & -\frac{2\tau\alpha}{H^2}e^{\boldsymbol{i}\lambda_m} \end{bmatrix}, \hat{\boldsymbol{\mathcal{M}}}_m^H(\boldsymbol{\theta}) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Comparison of convergence rates for two coarsest grid solvers. Once we have an estimate of the convergence rate $\mathcal{P}$, we can compute the number of iterations l

required to reach a desired tolerance 10^{-k} using the following

$$l = -\frac{k\ln(10)}{\ln(\mathcal{P})}.$$

From Tables 4.1 and 4.2 we see our direct solver converges much faster than the Chumchob-Chen AOS solver on several different coarsest grid sizes for both Lung CT and Hand examples (Examples 2 and 3 in Figure 4.3) respectively, especially on the 4×4 and 8×8 grids. This improvement has a significant impact on the number of iterations required to reach a desired tolerance, which in turn will have a noticeable effect on the number of FAS-NMG cycles needed to obtain a good registration result in addition to the time taken. As is also clear from both tables, the rates are too high and both solvers are not effective on the less coarse 16×16 grid, possibly due to limitation of the analysis. We conclude the coarsest grid should be kept as 8×8.

Hence our improved NMG method, to be denoted by **unconstrained INMG**, is taken as Algorithm 6 equipped with the coarsest grid solver by Algorithm 7 and the predicted number of smoothing steps of $\nu_1, \nu_2 \geq 8$ since

$$\mu_{\max}^8 = 0.74762^8 < 0.1$$

is believed to be sufficient.

4.4 Non-folding constraint model

We now present another model, this time with the aim of delivering diffeomorphic transforms. Folding in the transformation is a problem which can occur in image registration, unless it is specifically controlled. In real applications the presence of folding would suggest an inaccurate registration result as such transformations are non-physical. In this section we first introduce our proposed improved diffusion model, which removes any folding which may occur in the transformation $\boldsymbol{\varphi}$, in addition to including a NMG scheme (Algorithm 6). Next we extend this model further to increase robustness with respect to the choice of weighting parameter α and folding severity.

4.4.1 Improved diffusion model formulation and optimise-discretise approach

In the work by Burger et al. [18], it was explained the sign of the determinant $\det\left(\nabla\boldsymbol{\varphi}\right)$ can indicate the presence of any folding in the transformation $\boldsymbol{\varphi}$. Or more specifically the sign of

$$\det\left(\nabla\boldsymbol{\varphi}\right) = \left(1 + u_{1_{x_1}}\right)\left(1 + u_{2_{x_2}}\right) - u_{1_{x_2}} u_{2_{x_1}}. \tag{4.39}$$

If (4.39) ≤ 0 this indicates folding in the transformation is present, while if (4.39) > 0 no folding occurs in the transformation. In [18] this information was used to add an additional term into the diffusion energy functional (4.4) which penalises this determinant in order to produce diffeomorphic image registrations, thus resulting in the following 2D hyper-elastic energy functional

$$E^{Hyper}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega} \left|T_{\boldsymbol{u}} - R\right|^2 + \alpha \sum_{s=1}^{2} \left|\nabla u_s\right|^2 + \beta \left(\frac{\left(\det\left(\nabla \boldsymbol{\varphi}\right) - 1\right)^2}{\det\left(\nabla \boldsymbol{\varphi}\right)} \right)^2 d\Omega \tag{4.40}$$

where

$$\alpha \in \mathbb{R}^+,\, 0 \leq \beta \in \mathbb{R}$$

are weighting parameters. Although it may be possible to develop an effective smoother for solving (4.40), which has a strong non-linearity, we however propose an extension to the diffusion model (4.4) as a simplification of the hyper-elastic model (4.40) to control any folding. We propose to introduce a constraint into the diffusion model which ensures a positive value of the determinant (4.39). In other words, we aim to solve the following minimisation problem

$$\min_{\boldsymbol{u}} \left\{ E^{Diff}(\boldsymbol{u}) \right\},\ s.t.\ \det\left(\nabla \boldsymbol{\varphi}\right) > 0.$$

Or equivalently, using an optimise-discretise approach, we look to solve the following EL equations

$$-\alpha \Delta u_m + F_m(\boldsymbol{u}) = 0,\ s.t.\ \det\left(\nabla \boldsymbol{\varphi}\right) > 0 \tag{4.41}$$

with Neumann boundary conditions

$$\nabla u_m \cdot \boldsymbol{n} = 0$$

and where $F_m(\boldsymbol{u})$ are as in (4.7) for $m = 1, 2$.

4.4.2 Estimating the determinant using finite elements

In order for us to be able to impose the constraint in (4.41), we must first obtain an approximation of the determinant at every discrete interior point of Ω^h. In other words we need to compute

$$\begin{aligned} \boldsymbol{Q}^h \equiv \left(Q^h_{ij}\right) &= \left(\det\left(\nabla^h \boldsymbol{\varphi}^h\right)\right)_{i,j} \\ &= \left(1 + (\partial^h_{x_1} u^h_1)_{i,j}\right)\left(1 + (\partial^h_{x_2} u^h_2)_{i,j}\right) - \left(\partial^h_{x_2} u^h_1\right)_{i,j}\left(\partial^h_{x_1} u^h_2\right)_{i,j} \end{aligned}$$

where

$$\boldsymbol{Q}^h \in \mathbb{R}^{(n-2)\times(n-2)}$$

is the matrix consisting of determinant values at each of the discrete interior points (i,j) for $i,\, j = 2,\ldots,n-1$. To compute the entry $\left(Q^h_{ij}\right)$, we need to determine the discrete partial derivatives

$$(\partial^h_{x_1} u^h_m)_{i,j},\, (\partial^h_{x_2} u^h_m)_{i,j}$$

for $m = 1, 2$. We do this by splitting our discrete domain Ω^h into a mesh of finite elements consisting of piecewise linear triangular basis functions as shown in Figure 4.2(a). In fact for each interior point (i,j), we need to compute the determinant in each of the four triangles $T_1,\ldots,T_4$ as shown in Figure 4.2(b). Doing this gives us a clearer picture of the local geometry surrounding the (i,j) point, thus allowing us to better detect any mesh folding in the transformation. Once we have determinant values for each of the triangles, we assign the smallest value to be our $\left(Q^h_{ij}\right)$ entry, this in essence considers the worst possible case for each (i,j) allowing us to better correct all potential folding in the transformation. Now for linear triangular basis functions, we can approximate $u^h_m(\boldsymbol{x})$ by the following linear functions

$$L^h_m(\boldsymbol{x}) = r^h_{u_m} + s^h_{u_m} x_1 + t^h_{u_m} x_2 \tag{4.42}$$

where

$$r^h_{u_m},\, s^h_{u_m},\, t^h_{u_m} \in \mathbb{R}$$

are coefficients to be determined for $m = 1, 2$. From (4.42) we see the partial derivatives

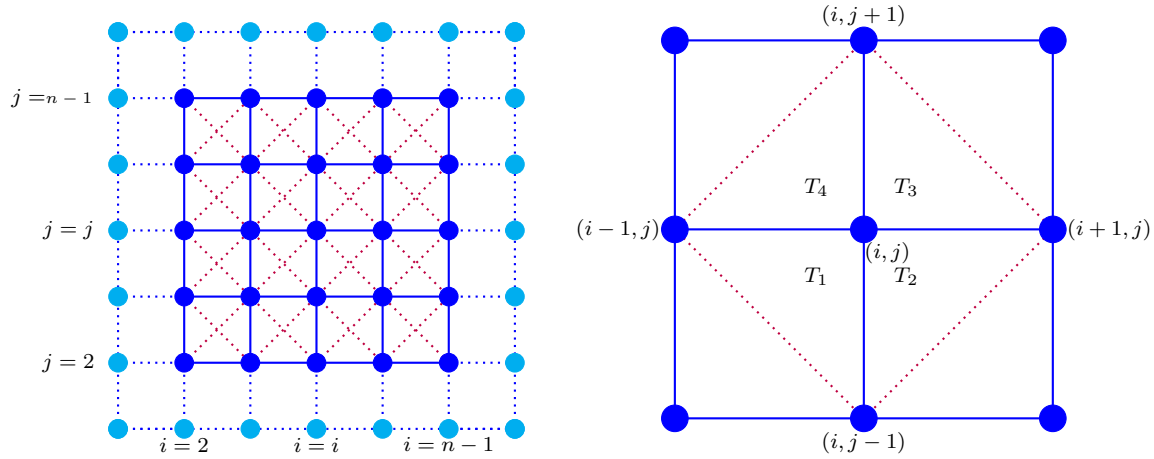
$$\partial^h_{x_1} u^h_m,\, \partial^h_{x_2} u^h_m$$

are given by the coefficients

$$s^h_{u_m},\, t^h_{u_m}$$

respectively. Then looking at the first triangle T_1, at a general discrete interior point (i,j), we have the following system

$$\textbf{Triangle } \boldsymbol{T_1}. \quad \begin{bmatrix} 1 & x_i & y_j \\ 1 & x_{i+1} & y_j \\ 1 & x_i & y_{j+1} \end{bmatrix} \begin{bmatrix} r^h_{1\,u_1} \\ s^h_{1\,u_1} \\ t^h_{1\,u_1} \end{bmatrix} = \begin{bmatrix} \left(u^h_1\right)_{i,j} \\ \left(u^h_1\right)_{i+1,j} \\ \left(u^h_1\right)_{i,j+1} \end{bmatrix},$$
$$\begin{bmatrix} 1 & x_i & y_j \\ 1 & x_{i+1} & y_j \\ 1 & x_i & y_{j+1} \end{bmatrix} \begin{bmatrix} r^h_{1\,u_2} \\ s^h_{1\,u_2} \\ t^h_{1\,u_2} \end{bmatrix} = \begin{bmatrix} \left(u^h_2\right)_{i,j} \\ \left(u^h_2\right)_{i+1,j} \\ \left(u^h_2\right)_{i,j+1} \end{bmatrix}; \tag{4.43}$$



(a) Global mesh of finite elements of the full discrete domain as shown by the purple dashed lines.

(b) Local mesh of finite elements at a general discrete point (i, j).

Figure 4.2: Finite element splitting of the discrete domain Ω^h using linear triangle basis functions.

For the remaining triangles T_2, T_3 and T_4, we can obtain similar systems to (4.43). Then, to compute the coefficients

$$r^h_{l\,u_m},\ s^h_{l\,u_m},\ t^h_{l\,u_m}$$

we solve

$$\boldsymbol{a}^h_l = \left[\boldsymbol{C}^h_{1\,l}\right]^{-1} \boldsymbol{v}^h_{1\,l}, \quad \boldsymbol{b}^h_l = \left[\boldsymbol{C}^h_{2\,l}\right]^{-1} \boldsymbol{v}^h_{2\,l} \tag{4.44}$$

where

$$\boldsymbol{C}^h_{1\,l} = \left[r^h_{l\,u_1}, s^h_{l\,u_1}, t^h_{l\,u_1}\right]^T,\ \boldsymbol{C}^h_{2\,l} = \left[r^h_{l\,u_2}, s^h_{l\,u_2}, t^h_{l\,u_2}\right]^T$$

are the column vectors of coefficients for $\left(u^h_1\right)_{i,j}$, $\left(u^h_2\right)_{i,j}$ respectively, $\left[\boldsymbol{C}^h_{*\,l}\right]^{-1}$ are the inverses of the matrices corresponding to the edges of the triangle T_l and

$$\boldsymbol{v}^h_{m\,l} = \left[u^h_{m\,1}, u^h_{m\,2}, u^h_{m\,3}\right]^T$$

are the values of u^h_m at each vertex of the triangle T_l for $l = 1, \ldots, 4$, $m = 1, 2$. Then, once all elements of $\boldsymbol{Q}^h$ have been computed, we take the minimum value of the matrix $\boldsymbol{Q}^h$ to be used to see if the constraint has been satisfied. This method can be summarised by Algorithm 8. Once we have a value for Q^h_{min}, we use Algorithm 9 to impose the constraint and determine whether we accept the updated transformation or not. Supposing the constraint in (4.41) is not satisfied once the solution $\boldsymbol{u}$ has been found, then we take a factor $\omega \in (0, 1)$ of the displacement and recalculate the constraint to

Algorithm 8 $Q^h_{min} \leftarrow ComputeQ(\boldsymbol{u}^h, \boldsymbol{n}, \boldsymbol{h})$

for $i = 2, \ldots, n-1$ **do**
  for $j = 2, \ldots, n-1$ **do**
    for $l = 1, \ldots, 4$ **do**
      Compute the vectors $\boldsymbol{a}^h_l$, $\boldsymbol{b}^h_l$ using (4.44)
      Compute determinant for triangle $T_l \rightarrow \tilde{Q}^h_l = (1 + s^h_{l\,u_1})(1 + t^h_{l\,u_2}) - t^h_{l\,u_1} s^h_{l\,u_2}$
    end for
    Assign minimum $\tilde{Q}^h$ to be entry $\left(Q^h_{ij}\right) \rightarrow \left(Q^h_{ij}\right) = \min\{\tilde{Q}^h_1, \ldots, \tilde{Q}^h_4\}$
  end for
end for
  Take minimum entry in $\boldsymbol{Q}^h$ to be minimum determinant value $\rightarrow Q^h_{min} = \min\{\boldsymbol{Q}^h\}$

Algorithm 9 $\boldsymbol{u}^{(l+1)}_h \leftarrow ConstrainU(\boldsymbol{u}^{(l)}_h, \boldsymbol{h}, \omega, LMAX)$

for $l = 1, \cdots, LMAX$ **do**
  Compute minimum value of determinant Q^h_{min} using Algorithm 8
  if $Q^h_{min} > 0$ *and* $l \leq LMAX$ **then**
    Accept update $\boldsymbol{u}^{(l+1)}_h = \boldsymbol{u}^{(l)}_h$
  else if $Q^h_{min} \leq 0$ *and* $l < LMAX$ **then**
    Reject update and set $\boldsymbol{u}^{(l)}_h = \omega \boldsymbol{u}^{(l)}_h$, $\omega \in (0, 1)$
  else if $Q^h_{min} \leq 0$ *and* $l = LMAX$ **then**
    Error $\rightarrow$ Constraint failed
  end if
end for

Algorithm 10 $\left[\boldsymbol{u}^{(l+1)}_h, c, done\_alpha\right] \leftarrow AdaptiveU(\boldsymbol{u}^{(l)}_h, \boldsymbol{h}, \omega, LMAX)$

Save current 'good' approximation $\rightarrow \hat{\boldsymbol{u}}^{(l)}_h = \boldsymbol{u}^{(l)}_h$, $c = 0$
for $l = 1, \cdots, LMAX$ **do**
  Compute minimum value of determinant Q^h_{min} using Algorithm 8
  if $Q^h_{min} > 0$ *and* $l \leq LMAX$ **then**
    Accept update $\boldsymbol{u}^{(l+1)}_h = \boldsymbol{u}^{(l)}_h$, $\hat{\boldsymbol{u}}^{(l)}_h = \boldsymbol{u}^{(l)}_h$, $c = c + 1$, $done\_alpha = 1$, *break*
  else if $Q^h_{min} \leq 0$ *and* $l < LMAX$ **then**
    Reject update and set $\boldsymbol{u}^{(l)}_h = \omega \boldsymbol{u}^{(l)}_h$, $\omega \in (0, 1)$, $c = c + 1$
  else if $Q^h_{min} \leq 0$ *and* $l = LMAX$ **then**
    Reset to 'good' approximation $\rightarrow c = LMAX$, $\boldsymbol{u}^{(l+1)}_h = \hat{\boldsymbol{u}}^{(l)}_h$, $done\_alpha = 0$
  end if
end for

see if has been satisfied. This process is performed a small number of times, and if the constraint still has not been satisfied then we deem the method to have failed. An illustration of this process can be seen in Algorithm 9.

In practice, Algorithm 8 can be computationally expensive on larger grid sizes owing to the fact we must solve eight inverse problems at every discrete interior point in the discrete domain Ω^h, consequently this has a severe impact on the CPU time of the NMG scheme for our constrained model. In Appendix A, we demonstrate how Algorithm 8 can be optimised to significantly decrease CPU cost for each iteration of the determinant computation. The method outlined in Algorithm 14 is how we actually compute the determinant in practice, and the results shown in §4.5.2 are also obtained using this algorithm.

Algorithm 11 $\boldsymbol{u}_{level}^{(l+1)} \leftarrow ConstFASNMG(level, \mu, \boldsymbol{u}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 steps

$$\tilde{\boldsymbol{u}}_{level}^{(l)} \leftarrow Smooth(\boldsymbol{u}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_1)$$

2: Coarse grid correction

Compute the residual $\boldsymbol{r}_{level}^{(l)} = \boldsymbol{F}_{level} - \boldsymbol{\mathcal{N}}_{level}(\tilde{\boldsymbol{u}}_{level}^{(l)})$

Restrict residual and smooth approximations $\boldsymbol{r}_{level-1}^{(l)} = \mathcal{R}_{level}^{level-1}\boldsymbol{r}_{level}^{(l)}$, $\tilde{\boldsymbol{u}}_{level-1}^{(l)} = \mathcal{R}_{level}^{level-1}\tilde{\boldsymbol{u}}_{level}^{(l)}$

Set $level \rightarrow level - 1$

Compute RHS of coarse grid PDE $\boldsymbol{f}_{level-1} = \boldsymbol{r}_{level-1}^{(l)} + \boldsymbol{\mathcal{N}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(l)}$

Compute an approximation $\bar{\boldsymbol{u}}_{level-1}^{(l)}$ to the coarse grid PDE $\boldsymbol{\mathcal{N}}_{level-1}\tilde{\boldsymbol{u}}_{level-1}^{(l)} = \boldsymbol{r}_{level-1}^{(l)}$

Solve residual equation on coarse grid to obtain approximations $\bar{\boldsymbol{u}}_{level-1}^{(l)}$

3: **if** $level = 1$ **then**

Use a direct or fast iterative solver to obtain the high accuracy solutions $\boldsymbol{u}_{level-1}^{(l)}$

4: Use Algorithm 9 to determine whether update is accepted

5: **else** $level > 1$ Repeat the FAS-cycle procedure recursively to the next level using $\tilde{\boldsymbol{u}}_{level-1}^{(l)}$ as an initial approximation i.e.

$$\bar{\boldsymbol{u}}_{level}^{(l)} \leftarrow ConstFASNMG(level - 1, \mu, \tilde{\boldsymbol{u}}_{level-1}^{(l)}, \boldsymbol{\mathcal{N}}_{level-1}, \boldsymbol{F}_{level-1}, \nu_1, \nu_2)$$

6: **end if**

Compute the correction $\boldsymbol{e}_{level-1}^{(l)} = \boldsymbol{u}_{level-1}^{(l)} - \tilde{\boldsymbol{u}}_{level-1}^{(l)}$

Interpolate the correction to next fine grid level $\boldsymbol{e}_{level}^{(l)} = \mathcal{I}_{level-1}^{level}\boldsymbol{e}_{level-1}^{(l)}$

Update current grid level approximations using correction $\hat{\boldsymbol{u}}_{level}^{(l)} = \tilde{\boldsymbol{u}}_{level}^{(l)} + \boldsymbol{e}_{level}^{(l)}$

7: Post-smoothing step by performing ν_2 steps

$$\boldsymbol{u}_{level}^{(l+1)} \leftarrow Smooth(\hat{\boldsymbol{u}}_{level}^{(l)}, \boldsymbol{\mathcal{N}}_{level}, \boldsymbol{F}_{level}, \nu_2)$$

8: Use Algorithm 9 to determine whether update is accepted if on finest grid level Ω^h

4.4.3 Numerical solution and NMG algorithm for a constrained diffusion model

Based on our **unconstrained INMG** framework, we solve our constrained diffusion model by NMG. Adding a constraint, the same pointwise smoother as the one shown in §4.2.2 and the same coarsest grid solver as the one described in §4.3.2 are used. Then our proposed NMG algorithm is shown in Algorithm 10, which we denote **constrained INMG**.

4.4.4 An adaptive α constrained diffusion model

While our **constrained INMG** does ensure the deformations obtained are non-folding, in cases where folding is severe the deformation field $\boldsymbol{u}$ can be penalised so heavily the deformed template image $T_{\boldsymbol{u}}$ may have moved very little when compared with the original template image T thus leading to a poor registration accuracy. To overcome this problem we propose an extension to our **constrained INMG** model, whereby we re-initialise the NMG method using a larger value of α if the constraint has not been satisfied within a small number of iterations. To construct this adaptive α scheme, we modify the determinant check in Algorithm 9 as shown in Algorithm 10. From

Algorithm 12 $\boldsymbol{u}_h^{(l+1)} \leftarrow Adaptive\alpha\big(R^h, T^h, \boldsymbol{n}, \boldsymbol{h}, \boldsymbol{u}_h^{(l)}, \alpha, i_{max}^{\alpha}\big)$

Set $done\_NMG = 0$, $done\_alpha = 0$
while $done\_NMG \neq 1$ **do**
  if $i^{\alpha} = i_{max}^{\alpha}$ **then**

$$LMAX = 100$$

  end if
  while $done\_NMG \neq 1$ **do**
    Set previous 'good' approximation $\rightarrow \boldsymbol{u}_h^{(l)} = \hat{\boldsymbol{u}}_h^{(l)}$
    Perform FAS-NMG

$$\rightarrow \Big[\boldsymbol{u}_h^{(l+1)}, c\Big] \leftarrow AdaptFASNMG\big(R^h, T^h, \boldsymbol{n}, \boldsymbol{h}, level, \hat{\boldsymbol{u}}_h^{(l)}, \boldsymbol{\mathcal{G}}^h, \alpha, \nu_1, \nu_2\big)$$

    if $c \leq LMAX$ *and* $done\_alpha \neq 1$ **then**
      break
    end if
    if NMG convergence criteria satisfied **then**

$$done\_NMG = 1$$

    end if
  end while
  if $c \leq LMAX$ *and* $done\_alpha \neq 1$ **then**
    Set $\alpha = 2\alpha$, $i^{\alpha} = i^{\alpha} + 1$, $\boldsymbol{u}_h^{(l)} = \hat{\boldsymbol{u}}_h^{(l)}$
  end if
end while

Algorithm 10 we see if we reach the iteration limit $LMAX$, we exit out of the FAS-NMG algorithm and this is when we re-initialise the NMG with a larger weighting parameter α. This process can be summarised by Algorithm 12, and where the algorithm $AdaptFASNMG$ is the same as Algorithm 11 except now Algorithm 12 is used to check the constraint instead of Algorithm 9. Another advantage of the adaptive α scheme shown in Algorithm 12 is its robustness to the choice of parameter α. Even if the initial α is set too small such that severe folding would normally occur, because we keep re-initialising the problem with new values of α, we automatically find a pseudo-optimal α value where folding is avoided while maintaining registration accuracy. This robustness will be shown in the next section. Using the pointwise smoother from §4.2.2, and the coarsest grid solver from §4.3.2 along with Algorithm 12, then we denote our adaptive α model by **adaptive INMG**.

4.5 Experimental results

Here we present and compare the results of four models:

(i) The Chumchob-Chen NMG method from [33] (Algorithm 6), which we have denoted **CCNMG**;

(ii) Our improved NMG method (Algorithm 6), which is denoted by **unconstrained INMG**;

(iii) Our proposed constrained NMG method (Algorithm 10), which we denoted by **constrained INMG**;

(iv) Our adaptive NMG method (Algorithm 12), which we denote **adaptive INMG**.

First we demonstrate how our more accurate analysis of the smoothing rate, along with our new coarsest grid solver, impact the number of NMG cycles required for the method to converge when compared with the **CCNMG** method. In addition we also show how the improved convergence of our NMG method our **unconstrained INMG** method results in a significant decrease in CPU time, along with an improvement in the accuracy of the registration, when compared with the **CCNMG** method.

Second, we show how our **constrained INMG** method overcomes the issue of transformation folding while still maintaining good accuracy and CPU times compared with our **unconstrained INMG** method and the **CCNMG** method.

Third we show how our **adaptive INMG** method not only overcomes the problem of mesh folding while keeping a good level of accuracy and CPU times, but also how it can maintain these good transforms while being robust to parameter choice when compared with the other models.

To gain a quantitative measure of the accuracy of the NMG methods, we use structural similarity (SSIM) [138] in addition to the relative error given by

$$Err = \frac{|T_{\boldsymbol{u}} - R|_2^2}{|R|_2^2}.$$

Moreover, in order to highlight the convergence problem of the **CCNMG** method, and for fairness, we consider a method to have converged only if one of the following stopping criteria has been satisfied:

(i) The average relative residual reaches a tolerance of $\varepsilon_1 = 10^{-2}$;

(ii) The maximum relative residual reaches a tolerance of $\varepsilon_2 = 10^{-2}$;

(iii) Maximum number of NMG cycles reaches $\varepsilon_3 = 25$.

We take 3 pairs of test images (shown in Figure 4.3) to experiment and compare registrations:

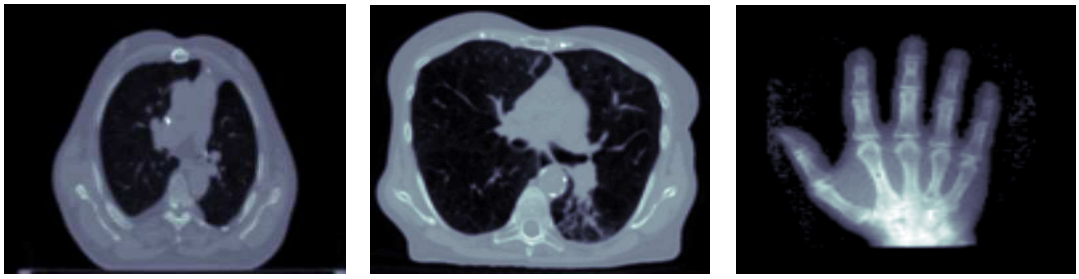
(i) **Example 1**. A pair of CT images from Figure 4.3(a, d);

(ii) **Example 2**. A second pair of CT images from Figure 4.3(b, e);

(iii) **Example 3**. A pair of Hand images from Figure 4.3(c, f).

Moreover, in Tables 4.5-4.6 we indicate whether a test has been 'successful' (results highlighted in green) or whether it has 'failed' (results highlighted in red). We say a test has 'failed' if the maximum number of NMG cycles ε_3 has been reached, or if there

(a) Reference R of Example 1. (b) Reference R of Example 2. (c) Reference R of Example 3.



(d) Template T of Example 1. (e) Template T of Example 2. (f) Template T of Example 3.

Figure 4.3: Three pairs of test images.

is folding in the result i.e.

$$Q^h_{min} < 0.$$

Additionally bold values indicate the results which give the best SSIM and relative error values for each test.

4.5.1 Comparative results of the CCNMG method and our unconstrained INMG method

Here we demonstrate the improvement of our **unconstrained INMG** method over the **CCNMG** method. As mentioned in §4.3, our improvement is to overcome the convergence problem which was present in the former method.

Test on Example 1. For the first lung CT example visual differences between the models are small in Figure 4.4. The first two columns of Table 4.3 show several test results of varying resolutions and parameters α. There, abbreviations '$SSIM$', 'Err', 'NMG', 'CPU' represent the final structural similarity, final relative error, number of multigrid cycles performed and CPU time respectively.

We can see, from Table 4.3, our **unconstrained INMG** method is successful for all cases of α but the **CCNMG** method failed in several cases. On convergence alone, the **CCNMG** method is not as fast as our **unconstrained INMG** method since it takes a larger number of NMG cycles.

Test on Example 2. In the second lung CT example, although visual differences be-

tween the models are again small in Figure 4.5, in Table 4.4 we see our **unconstrained INMG** method is better than the **CCNMG** method in all indicators $SSIM$, Err, NMG, CPU for the first α value, but for the other two cases of α both models failed to give diffeomorphic maps due to

$$\det\left(\nabla^h \boldsymbol{\varphi}^h\right) < 0.$$

Test on Example 3. From Figure 4.6, we see our **unconstrained INMG** method produces visually similar deformed template images $T_{\boldsymbol{u}}^h$ and final error images $\left|T_{\boldsymbol{u}}^h - R^h\right|$ when compared with those obtained from the **CCNMG** method. When we look at Table 4.5 we see our **unconstrained INMG** method requires consistently fewer NMG cycles to produce these accurate results. In fact, the **CCNMG** method method almost always fails to converge within the allowed number ε_3 of NMG cycles to the required tolerances. This confirms our statements earlier on the convergence problem of the **CCNMG** method. Moreover, this also leads to a drastic improvement in CPU time, especially in the 512^2 and 1024^2 cases where the **CCNMG** method requires a much larger number of NMG cycles.

4.5.2 Comparative results of our unconstrained INMG method and our constrained INMG method

In §4.4 we introduced our **constrained INMG** method in order to prevent any folding from occurring in the transformation. This was achieved by ensuring

$$\det\left(\nabla^h \boldsymbol{\varphi}^h\right) > 0$$

for every discrete interior point in Ω^h. Here we present results comparing our **unconstrained INMG** method and our **constrained INMG** method to show how this constraint does indeed prevent folding while still maintaining good accuracy and CPU time using the same three examples from §4.5.1. The abbreviation 'Q_{min}' represents the minimum determinant value $\det\left(\nabla^h \boldsymbol{\varphi}^h\right)$. Here small '$Err$' means a small fitting error while $Q_{min}^h > 0$ implies a correct registration transformation.

Test on Example 1. From columns 2 and 3 of Table 4.3 we see our **unconstrained INMG** method always produces positive Q_{min} values; as a result we obtain the exact same results with our **constrained INMG** method with very small increases in CPU times owing to the constraint checking. This also translates to Figure 4.4 where we see all images look very similar visually.

Test on Example 2. From Table 4.4 we see our **constrained INMG** method has overcome the mesh folding problems of our **unconstrained INMG** method by positive Q_{min} values in all cases. In achieving this convergent non-folding result, the number of NMG cycles taken by our **constrained INMG** method is more than our

unconstrained INMG method. Although the CPU times in these cases also increase noticeably, we do however still see a reduction and consistency in the number of NMG cycles when compared with the **CCNMG** method. The CPU time increase could be reduced further by a more computationally efficient implementation of our algorithm to penalise the transformation only in regions where folding is present.

Test on Example 3. Here we see the exact same pattern as in Example 1 since our **unconstrained INMG** method produces positive determinant values in all cases and identical results to our **constrained INMG** method with small increases in CPU times as shown in Table 4.5, with improvements in all categories over the **CCNMG** method especially in convergence and CPU times.

4.5.3 Comparative results of our constrained INMG method and our adaptive INMG method

In addition to our **constrained INMG** method, in §4.4 we also introduced an extension to our **constrained INMG** method with the goal of being robust to parameter choice while maintaining a non-folding transformation. Here we consider a case where severe folding would occur and our **constrained INMG** method, while producing a non-folding deformation, performs poorly in terms of registration accuracy whereas our **adaptive INMG** method produces good registration accuracy while also avoiding folding.

From Table 4.6 we see although we obtain very good accuracy from our **unconstrained INMG** method, we also have severe folding in the transformations for all tests as indicated by the negative Q_{min} values. Looking at the results for our **constrained INMG** method we see the folding problem has been overcome and all Q_{min} values are now positive, however we also see we have lost the accuracy of the result with regard to error when compared with our **unconstrained INMG** method results, especially on the 128^2 and 256^2 images. Our **adaptive INMG** method on the other hand not only produces non-folding results like with our **constrained INMG** method, but also maintains a similar level of accuracy when compared with the results from our **unconstrained INMG** method. In addition we also see our **adaptive INMG** method achieves this with only a slight increase in CPU time when compared with those from our **unconstrained INMG** method, whilst being over twice as fast as our **constrained INMG** method. From Figure 4.7 we see visually there is a noticeable difference between the deformed template from our **constrained INMG** method compared with those from our **unconstrained INMG** method and our **adaptive INMG** method, especially in the error images.

4.5.4 Test on NMG efficiency and parameter robustness

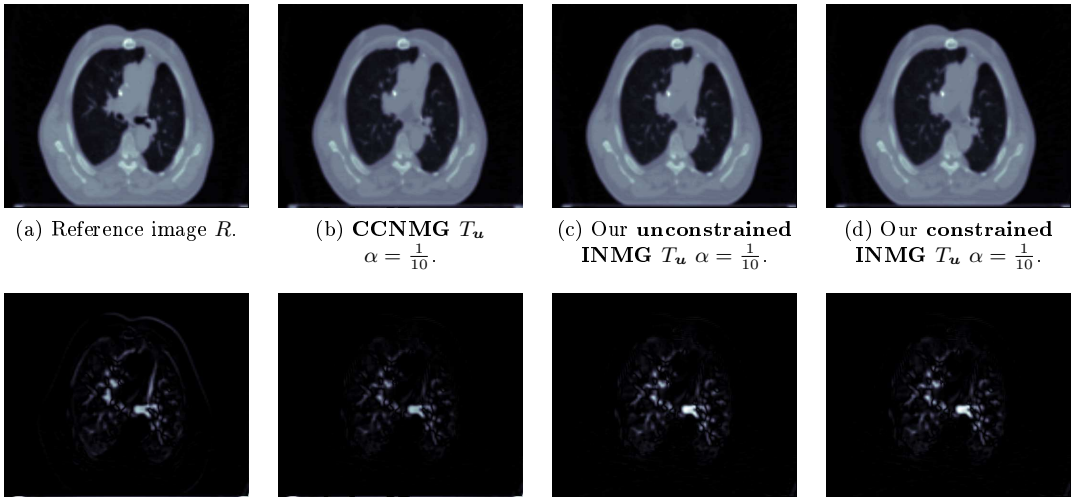
NMG efficiency. In this work, we are concerned with transformation quality and fast solution by a NMG. For the latter, we expect the optimal efficiency of $O(N \log N)$ complexity in achieving a fixed accuracy (with $N = n^2$ for $n \times n$ images). Then for an optimal NMG, we expect the CPU increase to be of ratio ≈ 4.5. In Table 5.11, we show test results of all four NMG methods for varying resolutions, where in the **CCNMG** method we use the original analysis of [33] to set the number of smoothing steps. Clearly our **unconstrained INMG** method, our **constrained INMG** method and our **adaptive INMG** method exhibit nearly optimal complexity but the **CCNMG** method shows irregular patterns, which justifies our re-analysis for Algorithm 6.

Finally to give an indication of the convergence history of the **CCNMG** method and our **unconstrained INMG** method, we plot in Figure 4.8 the residuals for more NMG cycles. Evidently our **unconstrained INMG** method has faster convergence plot than the **CCNMG** method.

Parameter robustness. In the diffusion model, the weighting parameter α indicates how strongly we wish to enforce smoothness on the deformation from the regularisation term. Specifically, a larger value of α will impose a strong penalisation on non-smooth deformations leading to no folding, however this also leads to a less accurate registration with regards to error. On the other hand, a smaller value of α leads to a more accurate registration in terms of error, but also increase the likelihood of folding occurring. Moreover, selecting a 'good' value for α can be very time consuming as in general a pre-multigrid routine is usually required to find this 'best' α (for example the cooling process in [33]), which can noticeably increase the computational work and CPU time. For this reason, having a model which is robust to the choice of weighting parameter is very useful as the need for finding the 'best' value for α is less important. Here we compare how the value of α impacts the relative error (denoted 'Err') and minimum determinant value (denoted 'Q_{min}') for our **unconstrained INMG** method and our **adaptive INMG** method. From Figure 4.9(a) we see as α gets smaller the error also decreases (as expected), however looking at Figure 4.9(b) we see the value of Q_{min} is also decreasing to a point where it is always negative (also as expected) as highlighted by the dotted line. This suggests our **unconstrained INMG** method has a limit where it maintains physically accurate non-folding deformations, and once past this point folding always occurs. Looking at Figure 4.10(a) we see our **adaptive INMG** method follows a similar pattern with regard to a decreasing error as α decreases like with our **unconstrained INMG** method, however from Figure 4.10(b) we see our **adaptive INMG** method always maintains the physical integrity of the deformation with $Q_{min} > 0$ for all tested values of α. From this we can conclude our **adaptive INMG** method is very robust to the initial value of α, even for small values, while maintaining a consistently good registration accuracy in terms of error.

4.6 Summary

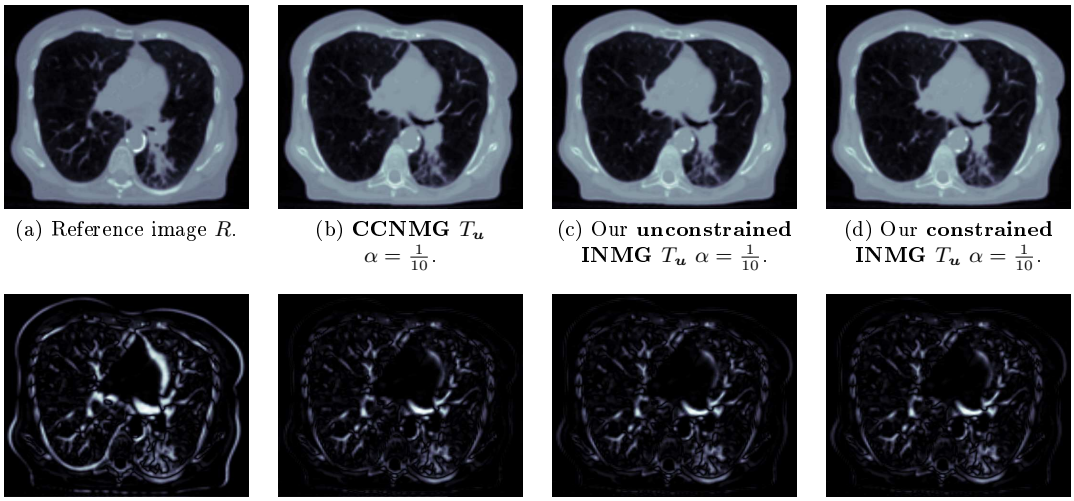
In this chapter we have first presented an improved NMG method, with regard to convergence and accuracy, over the one proposed by Chumchob and Chen through a more detailed and accurate analysis of the multigrid method, in addition to a different coarsest grid solver. Second we proposed an extension to our NMG method with the aim of producing non-folding transformations, which was achieved by imposing an additional constraint into our improved NMG method. Next we extended our **constrained INMG** to be more robust to parameter choice while keeping non-folding deformations and good registration accuracy. We then used three examples to demonstrate the improvement in accuracy and NMG cycles required for convergence over the Chumchob-Chen NMG, along with how our **constrained INMG** and **adaptive INMG** methods overcame folding by ensuring $\det\left(\nabla^h \boldsymbol{\varphi}^h\right) > 0$.



(a) Reference image R. (b) **CCNMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (c) Our **unconstrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (d) Our **constrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$.

(e) Initial $|T - R|$ $Err = 0.60\%$. (f) **CCNMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.43\%$. (g) Our **unconstrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.41\%$. (h) Our **constrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.41\%$.

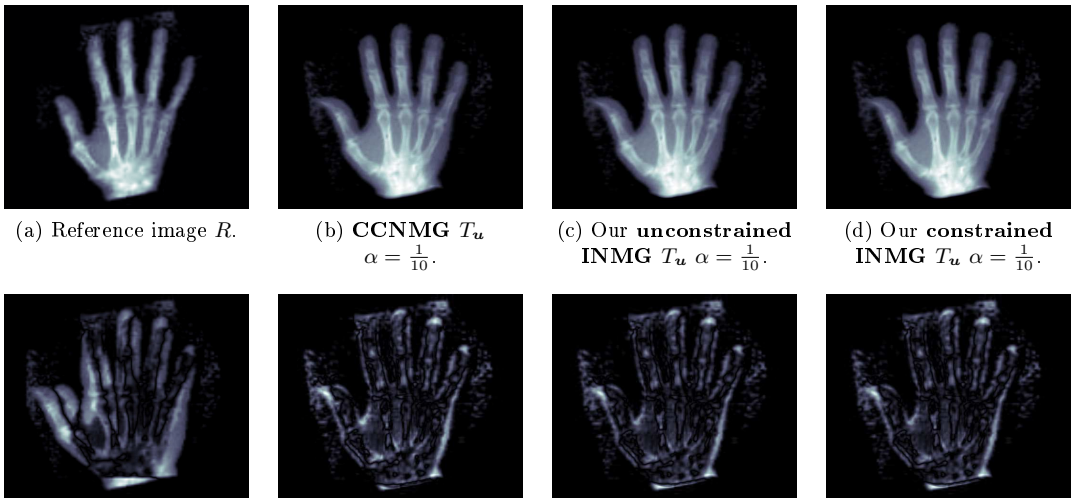
Figure 4.4: Example 1: Registration of 4.3(a) and 4.3(d) of size 512×512 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **CCNMG** model, our **unconstrained INMG** model and our **constrained INMG** model respectively, while images (f), (g) and (h) show the respective final errors.



(a) Reference image R. (b) **CCNMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (c) Our **unconstrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (d) Our **constrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$.

(e) Initial $|T - R|$ $Err = 1.99\%$. (f) **CCNMG** $|T_{\boldsymbol{u}} - R|$ $Err = 1.07\%$. (g) Our **unconstrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.95\%$. (h) Our **constrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.95\%$.

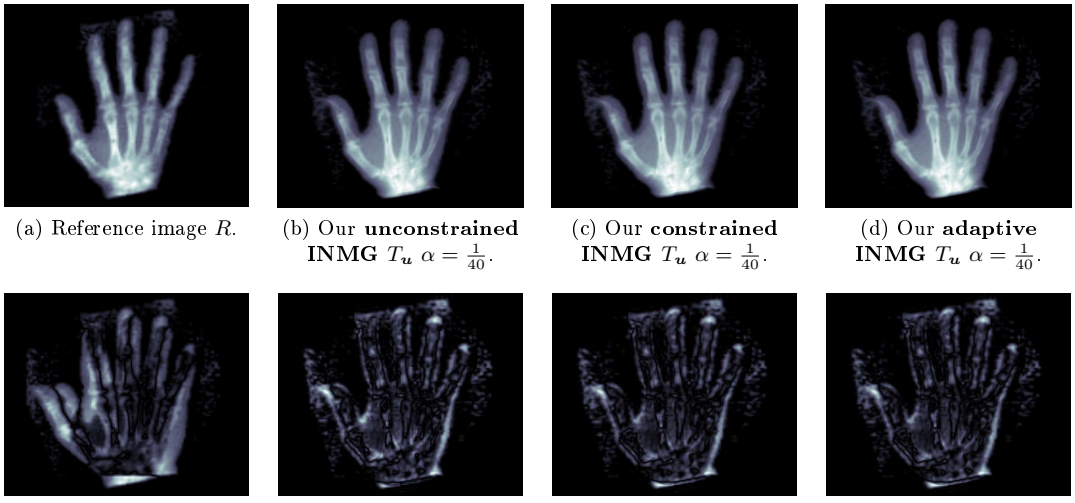
Figure 4.5: Example 2: Registration of 4.3(b) and 4.3(e) of size 512×512 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **CCNMG** model, our **unconstrained INMG** model and our **constrained INMG** model respectively, while images (f), (g) and (h) show the respective final errors.



(a) Reference image R. (b) **CCNMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (c) Our **unconstrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$. (d) Our **constrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{10}$.

(e) Initial $|T - R|$ $Err = 13.25\%$. (f) **CCNMG** $|T_{\boldsymbol{u}} - R|$ $Err = 2.02\%$. (g) Our **unconstrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 1.86\%$. (h) Our **constrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 1.86\%$.

Figure 4.6: Example 3: Registration of 4.3(c) and 4.3(f) of size 512×512 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **CCNMG** model, our **unconstrained INMG** model and our **constrained INMG** model respectively, while images (f), (g) and (h) show the respective final errors.



(a) Reference image R. (b) Our **unconstrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{40}$. (c) Our **constrained INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{40}$. (d) Our **adaptive INMG** $T_{\boldsymbol{u}}$ $\alpha = \frac{1}{40}$.

(e) Initial $|T - R|$ $Err = 13.25\%$. (f) Our **unconstrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.82\%$. (g) Our **constrained INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 1.10\%$. (h) Our **adaptive INMG** $|T_{\boldsymbol{u}} - R|$ $Err = 0.74\%$.

Figure 4.7: Example 3: Registration of 4.3(c) and 4.3(f) of size 512×512 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using our **unconstrained INMG** model, our **constrained INMG** model and our **adaptive INMG** model respectively, while images (f), (g) and (h) show the respective final errors for the bad parameter value $\alpha = \frac{1}{40}$ where severe folding occurs in the deformation.

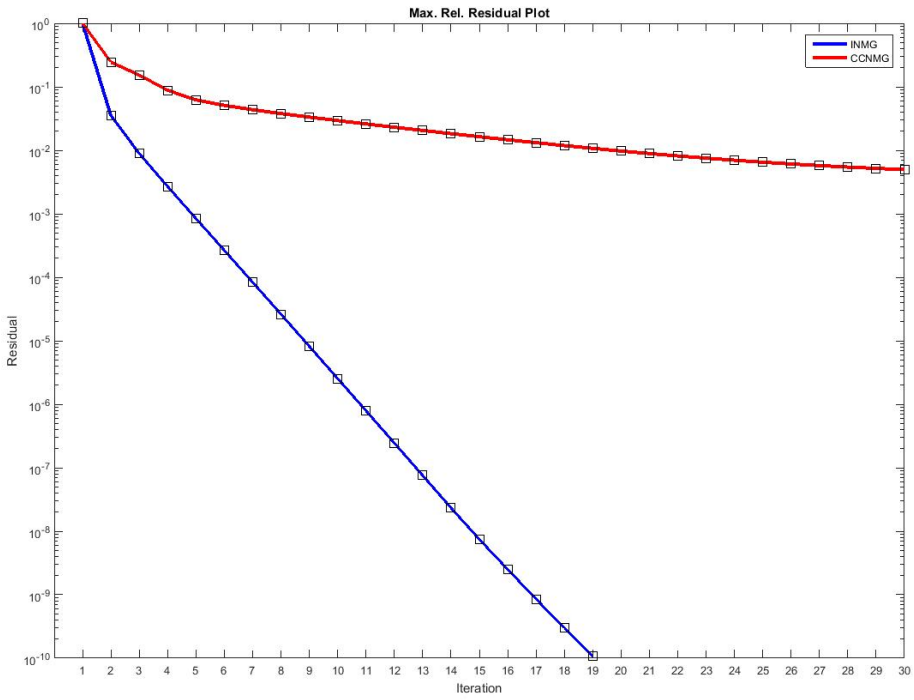
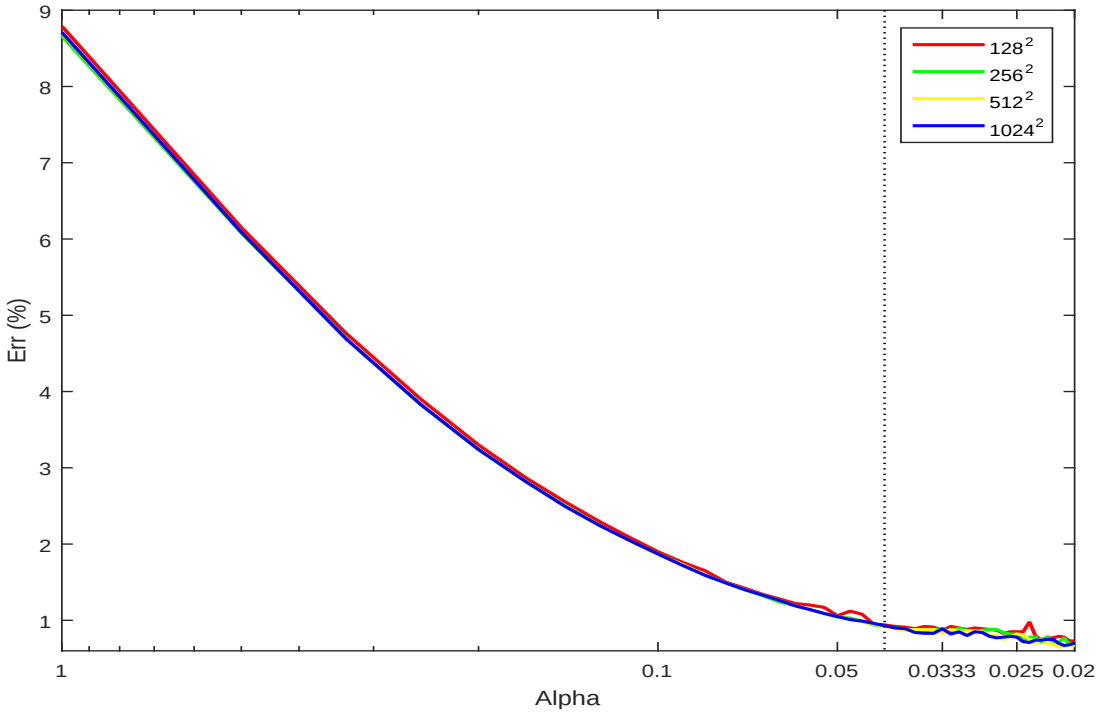
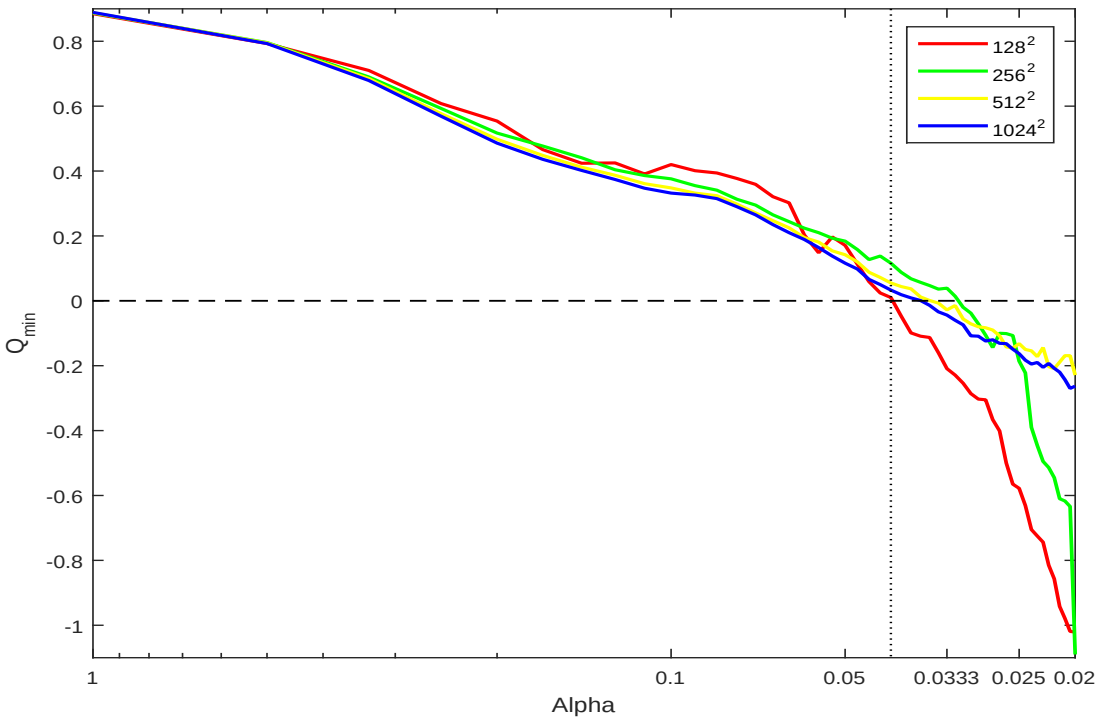


Figure 4.8: Comparison of the number of NMG cycles required for the maximum relative residual to reach a tolerance of 10^{-10} between our **unconstrained INMG** method and the **CCNMG** method.

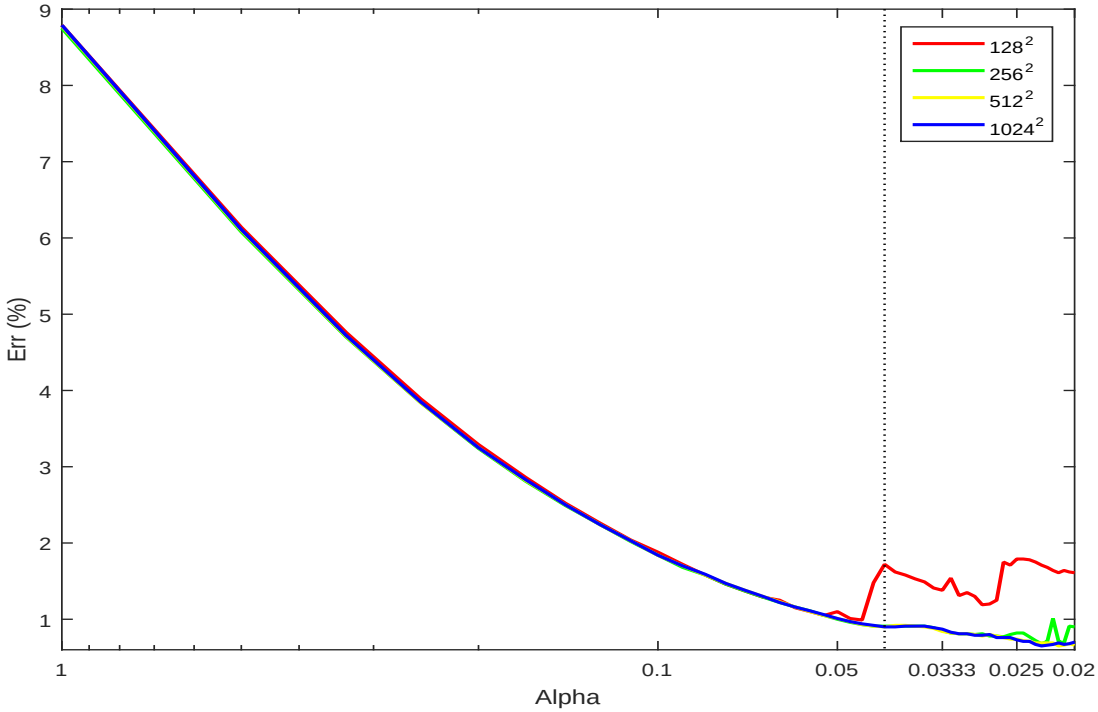


(a) Plot of relative error vs parameter α of our **unconstrained INMG** method for Example 3.

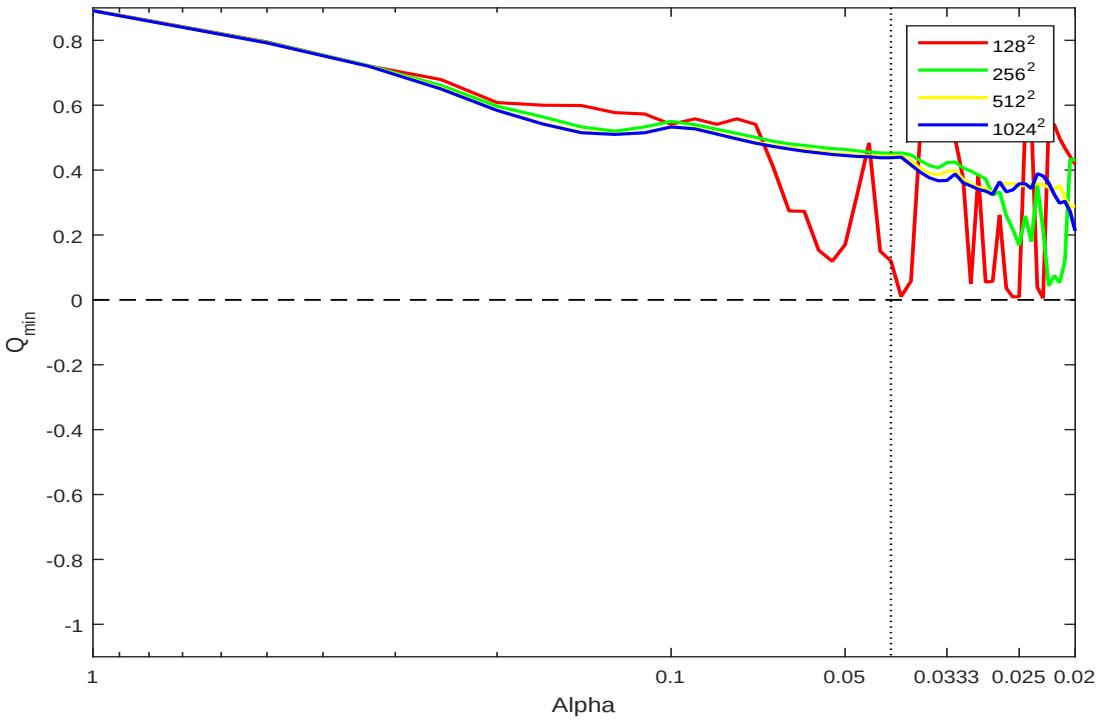


(b) Plot of minimum determinant value vs parameter α of our **unconstrained INMG** method for Example 3.

Figure 4.9: Test of the robustness of our **unconstrained INMG** method with respect to the choice of parameter α, for 50 parameter values.



(a) Plot of relative error vs parameter α of our **adaptive INMG** method for Example 3.



(b) Plot of minimum determinant value vs parameter α of our **adaptive INMG** method for Example 3.

Figure 4.10: Test of the robustness of our **adaptive INMG** method with respect to the choice of parameter α, for 50 parameter values.

		CCNMG					unconstrained INMG					constrained INMG				
Image Size (n^2)	α	***SSIM***	***Err* (%)**	***NMG***	***CPU* (*s*)**	Q_{min}	***SSIM***	***Err* (%)**	**NMG**	**CPU (s)**	Q_{min}	***SSIM***	***Err* (%)**	**NMG**	**CPU (s)**	Q_{min}
128^2		0.930	0.54	2	0.391	0.797	**0.943**	**0.41**	1	0.333	0.819	**0.943**	**0.41**	1	0.439	0.819
256^2	$\frac{1}{5}$	0.943	0.45	5	1.512	0.715	**0.951**	**0.42**	2	1.927	0.803	**0.951**	**0.42**	2	2.051	0.803
512^2		0.959	0.44	13	22.387	0.854	**0.964**	**0.43**	2	9.426	0.801	**0.964**	**0.43**	2	9.721	0.801
1024^2		0.972	0.44	25	196.585	0.872	**0.975**	**0.43**	3	66.178	0.822	**0.975**	**0.43**	3	69.500	0.822
128^2		0.931	0.52	1	0.316	0.612	**0.945**	**0.39**	1	0.425	0.694	**0.945**	**0.39**	1	0.437	0.694
256^2	$\frac{1}{10}$	0.945	0.43	25	6.887	0.464	**0.953**	**0.40**	1	1.090	0.660	**0.953**	**0.40**	1	1.164	0.660
512^2		0.961	0.43	10	17.204	0.734	**0.965**	**0.41**	1	5.057	0.668	**0.965**	**0.41**	1	5.250	0.668
1024^2		0.974	0.43	23	180.785	0.745	**0.976**	**0.42**	1	22.972	0.685	**0.976**	**0.42**	1	24.182	0.685
128^2		0.937	0.45	25	1.919	0.619	**0.947**	**0.38**	3	0.976	0.559	**0.947**	**0.38**	3	1.010	0.559
256^2	$\frac{1}{15}$	0.948	0.40	25	6.820	0.230	**0.954**	**0.39**	1	1.080	0.511	**0.954**	**0.39**	1	1.146	0.511
512^2		0.962	0.41	12	20.657	0.631	**0.966**	**0.40**	1	4.886	0.526	**0.966**	**0.40**	1	5.150	0.526
1024^2		0.975	0.41	18	141.395	0.644	**0.977**	**0.40**	1	24.642	0.554	**0.977**	**0.40**	1	25.546	0.554

Table 4.3: Example 1. Registration comparison of three methods on multiple image sizes for different α values, with an initial relative error of 0.60% and initial SSIM values of 0.933, 0.942, 0.957, 0.972 for the 128^2, 256^2, 512^2, 1024^2 images respectively.

		CCNMG					unconstrained INMG					constrained INMG				
Image Size (n^2)	α	***SSIM***	***Err* (%)**	***NMG***	***CPU* (*s*)**	Q_{min}	***SSIM***	***Err* (%)**	**NMG**	**CPU (s)**	Q_{min}	***SSIM***	***Err* (%)**	**NMG**	**CPU (s)**	Q_{min}
128^2		0.750	1.28	4	0.530	0.642	**0.764**	**1.17**	2	0.586	0.664	**0.764**	**1.17**	2	0.603	0.664
256^2	$\frac{1}{5}$	0.752	1.35	11	3.102	0.640	**0.786**	**1.14**	3	2.926	0.645	**0.786**	**1.14**	3	3.015	0.645
512^2		0.806	1.32	25	42.794	0.618	**0.832**	**1.18**	4	18.561	0.683	**0.832**	**1.18**	4	19.188	0.683
1024^2		0.860	1.34	25	199.920	0.640	**0.883**	**1.20**	4	89.853	0.701	**0.883**	**1.20**	4	94.397	0.701
128^2		0.766	1.04	3	0.456	0.406	**0.783**	**0.95**	2	0.636	0.070	**0.783**	**0.95**	2	0.715	0.070
256^2	$\frac{1}{10}$	0.768	1.11	7	2.038	0.344	**0.803**	**0.91**	3	2.879	−0.028	0.800	0.95	6	6.251	0.027
512^2		0.819	1.07	20	34.047	0.280	**0.847**	**0.95**	3	14.244	0.091	**0.847**	**0.95**	3	14.784	0.091
1024^2		0.873	1.06	25	195.431	0.271	**0.893**	**0.96**	4	68.196	0.145	**0.893**	**0.96**	4	71.186	0.145
128^2		0.774	0.89	3	0.488	0.080	**0.791**	**0.81**	3	0.920	−0.687	0.757	1.18	8	3.424	0.015
256^2	$\frac{1}{15}$	0.802	0.77	6	1.786	−0.165	**0.811**	**0.76**	2	1.952	−0.862	0.772	1.23	5	8.047	0.024
512^2		0.826	0.91	15	25.598	−0.122	**0.854**	**0.79**	3	13.750	−0.680	0.827	1.18	6	40.789	0.012
1024^2		0.880	0.89	25	195.370	−0.156	**0.899**	**0.80**	3	69.076	−0.584	0.881	1.16	6	182.460	0.011

Table 4.4: Example 2. Registration comparison of three methods on multiple image sizes for different α values, with an initial relative error of 1.99% and initial SSIM values of 0.667, 0.704, 0.769, 0.838 for the 128^2, 256^2, 512^2, 1024^2 images respectively.

		CCNMG					unconstrained INMG					constrained INMG				
Image Size (n^2)	α	***SSIM***	***Err*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err*** (%)	**NMG**	**CPU** (s)	Q_{min}	***SSIM***	***Err*** (%)	**NMG**	**CPU** (s)	Q_{min}
128^2		**0.742**	**2.42**	16	1.464	0.665	0.717	3.30	2	0.633	0.554	0.717	3.30	2	0.644	0.554
256^2	$\frac{1}{5}$	**0.743**	**2.61**	25	7.017	0.701	0.725	3.24	2	1.959	0.517	0.725	3.24	2	2.093	0.517
512^2		0.748	3.68	25	45.542	0.717	**0.750**	**3.24**	2	9.397	0.498	**0.750**	**3.24**	2	9.691	0.498
1024^2		0.747	6.85	25	195.731	0.648	**0.784**	**3.24**	2	45.445	0.486	**0.784**	**3.24**	2	47.728	0.486
128^2		**0.775**	**1.46**	10	0.959	0.600	0.758	1.89	3	0.868	0.420	0.758	1.89	3	0.892	0.420
256^2	$\frac{1}{10}$	**0.776**	**1.46**	25	6.787	0.639	0.760	1.87	2	1.984	0.376	0.760	1.87	2	2.118	0.376
512^2		**0.778**	2.02	25	42.149	0.602	**0.778**	**1.86**	2	9.350	0.348	**0.778**	**1.86**	2	9.706	0.348
1024^2		0.780	3.63	25	195.403	0.532	**0.807**	**1.87**	2	45.620	0.332	**0.807**	**1.87**	2	48.026	0.332
128^2		**0.790**	**1.13**	8	0.814	0.563	0.783	1.33	3	0.891	0.324	0.783	1.33	3	0.922	0.324
256^2	$\frac{1}{15}$	**0.791**	**1.10**	22	5.992	0.561	0.781	1.31	3	2.907	0.266	0.781	1.31	3	3.086	0.266
512^2		0.786	1.40	25	42.225	0.539	**0.794**	**1.31**	3	13.786	0.246	**0.794**	**1.31**	3	14.526	0.246
1024^2		0.789	2.36	25	194.026	0.390	**0.819**	**1.31**	3	66.949	0.235	**0.819**	**1.31**	3	69.405	0.235

Table 4.5: Example 3. Registration comparison of three methods on multiple image sizes for different α values, with an initial relative error of 13.25% and initial SSIM values of 0.551, 0.587, 0.639, 0.693 for the 128^2, 256^2, 512^2, 1024^2 images respectively.

		unconstrained INMG					constrained INMG					adaptive INMG				
Image Size (n^2)	α	***SSIM***	***Err*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err*** (%)	**NMG**	**CPU** (s)	Q_{min}	***SSIM***	***Err*** (%)	**NMG**	**CPU** (s)	Q_{min}
128^2		**0.812**	**0.95**	2	0.686	−3.078	0.630	7.56	6	2.676	0.032	0.758	1.91	3	0.711	0.554
256^2	$\frac{1}{40}$	**0.816**	**0.74**	2	2.458	−0.463	0.630	9.59	3	5.076	0.060	0.815	0.82	2	2.178	0.168
512^2		**0.824**	0.82	2	9.729	−0.132	0.805	1.10	4	27.558	0.025	**0.824**	**0.74**	2	10.318	0.351
1024^2		0.832	0.78	2	45.762	−0.163	0.812	1.64	4	121.546	0.086	**0.842**	**0.73**	2	58.604	0.358

Table 4.6: Example 3. Registration comparison of three methods on multiple image sizes for a 'bad' choice of α, with an initial relative error of 13.25% and initial SSIM values of 0.551, 0.587, 0.639, 0.693 for the 128^2, 256^2, 512^2, 1024^2 images respectively.

			CCNMG		unconstrained INMG		constrained INMG		adaptive INMG	
Image Size (n^2)	**Image Example**	α	***CPU* (s)**	**Ratio**	***CPU* (s)**	**Ratio**	***CPU* (s)**	**Ratio**	***CPU* (s)**	**Ratio**
128^2			0.316	–	0.425	–	0.437	–	0.452	–
256^2			6.887	21.794	1.090	2.565	1.164	2.666	1.304	2.885
512^2	Example 1 (CT)	$\frac{1}{10}$	17.204	2.498	5.057	4.639	5.250	4.510	6.202	4.756
1024^2			180.785	10.508	22.972	4.543	24.182	4.606	29.072	4.688
128^2			0.456	–	0.636	–	0.715	–	0.831	–
256^2			2.038	4.469	2.879	4.527	6.251	8.743	3.874	4.662
512^2	Example 2 (CT)	$\frac{1}{10}$	34.047	16.706	14.244	4.948	14.784	2.365	18.768	4.845
1024^2			195.431	5.740	68.196	4.788	71.186	4.815	87.203	4.646
128^2			0.959	–	0.868	–	0.892	–	0.845	–
256^2			6.787	7.077	1.984	2.286	2.118	2.374	2.582	3.059
512^2	Example 3 (Hand)	$\frac{1}{10}$	42.149	6.210	9.350	4.713	9.706	4.089	12.340	4.779
1024^2			195.403	4.636	45.620	4.879	48.026	4.948	58.466	4.738

Table 4.7: Test on optimal complexity in CPU time ratio for four NMG methods. The optimal ratio is 4.5 for an $\mathcal{O}(N \log N)$ method (with $N = n^2$). Clearly the newer NMGs are better.

Chapter 5

An effective diffeomorphic model and its fast multigrid algorithm for the registration of lung CT images

A challenge which frequently arises in many real world applications, and especially in medical imaging, is image registration. An image registration technique works by fixing one image in a pair or set of similar images to be the 'reference' image and then applying geometric transformations to the remaining image/s called the 'template' image/s, with the goal of aligning the template image/s to the reference image. The important role which registration plays in many aspects of medical imaging problems can be seen in recent works of [2, 29, 60, 63, 76]. More specifically image registration is an important technique in diagnostics of lung problems [20, 30, 62, 65, 107, 119] where tasks such as motion correction and feature tracking are routinely carried out and any increase in accuracy is highly desirable in improving patient care. Since transformations within lung images are in general highly non-uniform, non-parametric models such as [12, 13, 15, 16, 18] are typically favoured over parametric models such as [6, 31, 82, 94]. Our main concern is this former type.

5.1 Introduction

Denoting by

$$R, T \in \Omega \subset \mathbb{R}^d$$

respectively a reference and template image function, we are looking to determine the transformation

$$\boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})$$

such that

$$T\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) \equiv T\big(\boldsymbol{x}+\boldsymbol{u}\big) \equiv T_{\boldsymbol{u}} \approx R \equiv R(\boldsymbol{x}) \text{ for } \boldsymbol{x} = \big[x_1, \ldots, x_d\big]^T \in \Omega \subset \mathbb{R}^d$$

where

$$\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}) = \big[u_1(\boldsymbol{x}), \ldots, u_d(\boldsymbol{x})\big]^T$$

denotes the displacement field. Throughout the remainder of this chapter we only consider the two-dimensional case $d = 2$, however the ideas presented are extendible to the three-dimensional case $d = 3$. In addition, we also assume the image domain Ω is given by the unit square $\Omega = [0, 1]^2$.

We can formulate the variational image registration problem mathematically in the following way. Rather than searching for the transformation $\boldsymbol{\varphi}$, we can equivalently determine the displacement field $\boldsymbol{u}$, which we achieve by solving a minimisation problem of the following form

$$\min_{\boldsymbol{u}} \big\{ E(\boldsymbol{u}) = \mathscr{D}\big(R, T, \boldsymbol{u}\big) + \alpha \mathscr{R}\big(\boldsymbol{u}\big) \big\} \tag{5.1}$$

where $E(\boldsymbol{u})$ denotes some general energy functional, $\mathscr{D}$ is some dissimilarity measure between R and T, $\mathscr{R}$ is a regularisation term required to constrain $\boldsymbol{u}$ and overcome the ill-posedness of the problem and $\alpha \in \mathbb{R}^+$ is some weighting parameter. In addition let us assume R, T are mono-modal images, then the common choice of dissimilarity measure is the sum of squared distances (SSD) (although this is not the only possible choice [101]) which is given by the following

$$\mathscr{D}\big(R, T, \boldsymbol{u}\big) = \frac{1}{2} \int_{\Omega} |T_{\boldsymbol{u}} - R|^2 \, d\Omega \tag{5.2}$$

where $|\cdot|$ denotes the Euclidean norm and

$$T_{\boldsymbol{u}} \equiv T(\boldsymbol{x} + \boldsymbol{u}).$$

Moreover, there are a large choice of regularisation terms which we can choose from [4, 11, 35, 51, 100], however we mainly consider the diffusion regulariser given by

$$\mathscr{R}(\boldsymbol{u}) = \frac{1}{2} \int_{\Omega} \sum_{s=1}^{2} \big|u_s\big|^2 \, d\Omega$$

in order to focus on the idea of diffeomorphism of $\boldsymbol{\varphi}$. Unfortunately energy functionals of the form shown in (5.1), in general, do not avoid the potential problem of mesh folding in the transformation $\boldsymbol{\varphi}$. Since we are considering real life medical imaging problems, a transformation with folding would suggest the transformation is physically inaccurate and therefore incorrect. One mathematical solution to overcome this problem is to

impose the non-linear constraint

$$Q_{min} = \min\big\{\det(\nabla\boldsymbol{\varphi})\big\} > 0$$

as done in recent works of [18,89,92,153] and in particular the term

$$\min\left\{\left(\frac{\big(\det(\nabla\boldsymbol{\varphi})-1\big)^2}{\det(\nabla\boldsymbol{\varphi})}\right)^2\right\}$$

added in [18].

We intend however, to consider another solution to this folding problem by extending the model (5.1) to include an additional term which explicitly links the forward transformation $\boldsymbol{\varphi}$ between T and R, and the backwards transformation $\boldsymbol{\psi}$ between R and T. Doing this enforces the transformation $\boldsymbol{\varphi}$ to be inverse consistent and therefore non-folding. A simple way to ensure diffeomorphism is for the transformation $\boldsymbol{\varphi}$ and the backwards transformation $\boldsymbol{\psi}$ to satisfy the relation

$$\boldsymbol{\varphi} = \boldsymbol{\psi}^{-1}$$

since

$$\big(\boldsymbol{\varphi}\circ\boldsymbol{\varphi}^{-1}\big)\boldsymbol{x} = \big(\boldsymbol{\psi}\circ\boldsymbol{\psi}^{-1}\big)\boldsymbol{x} = \boldsymbol{I}\boldsymbol{x} = \boldsymbol{x}$$

where $\boldsymbol{I}$ denotes the identity mapping. The first variant including an inverse consistency constraint on $\boldsymbol{\varphi}$ only, leads to a minimisation problem of the form

$$\min_{\boldsymbol{u}}\Big\{E^{(I)}(\boldsymbol{u}) = \mathscr{D}\big(R,T,\boldsymbol{u}\big) + \alpha\mathscr{R}\big(\boldsymbol{u}\big) + \beta\mathscr{I}\big(\boldsymbol{\varphi}(\boldsymbol{x}),\boldsymbol{\varphi}^{-1}(\boldsymbol{x})\big)\Big\} \tag{5.3}$$

where $\mathscr{I}$ denotes the inverse consistency constraint, $\boldsymbol{\varphi}^{-1}$, $\tilde{\boldsymbol{u}}$ denote the inverses of $\boldsymbol{\varphi}$, $\boldsymbol{u}$ respectively and $0 \leq \beta \in \mathbb{R}$ is a second weighting parameter. There are different choices for the inverse consistency constraint which can be seen in the works [26,28,30,82]. However we consider the second variant of an inverse consistent model, which uses both $\boldsymbol{\varphi}$ and $\boldsymbol{\psi}$, and has the following form

$$\begin{aligned}\min_{\boldsymbol{u},\boldsymbol{v}}\Bigg\{E^{(II)}\big(\boldsymbol{u},\boldsymbol{v}\big) = \frac{1}{2}\int_{\Omega}\mathscr{D}\big(R,T,\boldsymbol{u}\big) + \mathscr{D}\big(T,R,\boldsymbol{v}\big) + \alpha\big[\mathscr{R}\big(\boldsymbol{u}\big) + \mathscr{R}\big(\boldsymbol{v}\big)\big]\\ +\beta\Big[\mathscr{I}\big(\boldsymbol{\varphi}(\boldsymbol{x}),\boldsymbol{\psi}^{-1}(\boldsymbol{x})\big) + \mathscr{I}\big(\boldsymbol{\psi}(\boldsymbol{x}),\boldsymbol{\varphi}^{-1}(\boldsymbol{x})\big)\Big]\,d\Omega\Bigg\}.\end{aligned} \tag{5.4}$$

where

$$\mathscr{D}\big(T,R,\boldsymbol{v}\big),\ \mathscr{R}\big(\boldsymbol{v}\big),\ \mathscr{I}\big(\boldsymbol{\psi}(\boldsymbol{x}),\boldsymbol{\varphi}^{-1}(\boldsymbol{x})\big)$$

denote the similarity measure, regularisation term and inverse consistency constraint respectively for the backward problem $R \to T$. In addition $\boldsymbol{v}$, $\boldsymbol{\psi}$ denote the backward

displacement and transformation respectively with $\tilde{\boldsymbol{v}}$, $\boldsymbol{\psi}^{-1}$ denoting their inverses. We aim to simplify this second variant and propose an efficient multigrid numerical scheme.

The remainder of this chapter will be set out as followed. In §5.2 we introduce the Christensen-Johnson model [28] based on (5.4), in addition to our proposed simplification to avoid additional non-linearities when compared with general diffusion type models, in addition to our proposed numerical approach. Next in §5.3 we introduce our fast NMG scheme to overcome the increased computational cost resulting from the additional work required by the model, before showing some experimental results on real medical CT images in §5.4. Finally in §5.5 we present a summary of this chapter.

5.2 A simplified inverse consistent model and its algorithm

Several authors have discussed similar registration models for two images to symmetrically deform toward one another in multiple passes [26, 66, 113, 147]. The realisation of a diffeomorphic transform is achieved by working with four deformation fields instead of one. Here we follow the work by Christensen and Johnson [28] who proposed a model to overcome the problem of non-inverse consistent transformations by using only two deformation fields which is therefore computationally less complex. The model satisfies our requirement of having a more physically accurate transformation robust to folding, and was achieved through a combination of two things:

(i) A term was added into the standard form of the energy functional shown in (5.1) to impose inverse consistency and take on the form show in (5.4);

(ii) The forward ($T \to R$) and backward ($R \to T$) registration problems were computed simultaneously.

These things, combined with a SSD dissimilarity term (5.2) and diffusion regularisation term, led to the formation of their inverse consistent model which is given by the following

$$\min_{\boldsymbol{u},\boldsymbol{v}} \Bigg\{ E^{IC}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\int_{\Omega} |T_{\boldsymbol{u}} - R|^2 + |R_{\boldsymbol{v}} - T|^2 + \alpha \left[|\nabla \boldsymbol{u}|^2 + |\nabla \boldsymbol{v}|^2 \right] + \beta \left[\left| \boldsymbol{\varphi}(\boldsymbol{x}) - \boldsymbol{\psi}^{-1}(\boldsymbol{x}) \right|^2 + \left| \boldsymbol{\psi}(\boldsymbol{x}) - \boldsymbol{\varphi}^{-1}(\boldsymbol{x}) \right|^2 \right] d\Omega \Bigg\} \tag{5.5}$$

where $|\cdot|$ denotes the Frobenius norm for matrices and reduces to modulus for scalar quantities, $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$ denote the forward and backward transformations, $\boldsymbol{\varphi}^{-1}$, $\boldsymbol{\psi}^{-1}$ denote the inverse transformations, $\boldsymbol{u}$, $\boldsymbol{v}$ denote the forward and backward displacements and $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{v}}$ denote the inverse displacements respectively. An illustration of the various transformations can be seen in Figure 5.1 The full minimisation problem was then split into two sub-problems corresponding to the forward and backward registration problems

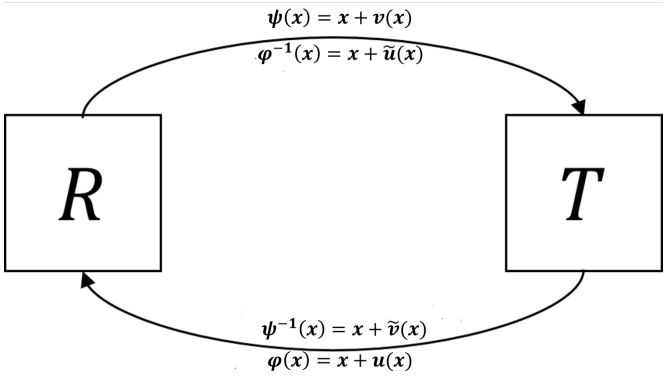
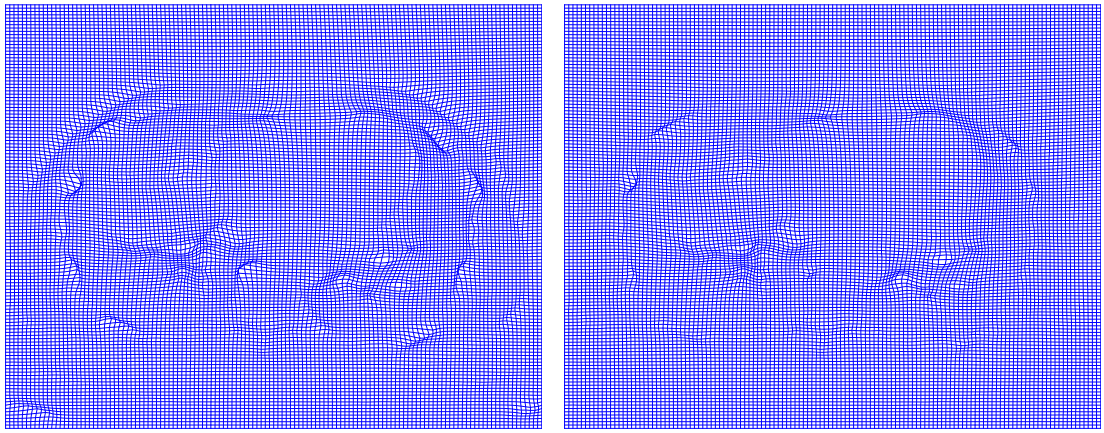


Figure 5.1: Illustration of the four transformations as seen in (5.5).



(a) Bad mesh of the transformation $\boldsymbol{\varphi}$ obtained from the standard diffusion model: $Q_{min} = -0.245$.

(b) Good mesh obtained from the new inverse consistent model: $Q_{min} = 0.114$.

Figure 5.2: Comparison of two registration meshes for Example 2 as shown in Figure 5.3 for the same parameters $\alpha = \frac{1}{25}$ and $\beta = 10^4$ (See §5.4).

respectively. This resulted in (5.5) being written in the following way

$$\begin{cases} \min\limits_{\boldsymbol{u}} \left\{ E_1^{IC}(\boldsymbol{u},\boldsymbol{v}) = \dfrac{1}{2}\displaystyle\int_\Omega |T_{\boldsymbol{u}} - R|^2 + \alpha\,|\nabla \boldsymbol{u}|^2 + \beta\,|\boldsymbol{u} - \tilde{\boldsymbol{v}}|^2 \; d\Omega \right\}, \; \tilde{\boldsymbol{v}}(\boldsymbol{x}) = \boldsymbol{\psi}^{-1}(\boldsymbol{x}) - \boldsymbol{x}, \\ \\ \min\limits_{\boldsymbol{v}} \left\{ E_2^{IC}(\boldsymbol{u},\boldsymbol{v}) = \dfrac{1}{2}\displaystyle\int_\Omega |R_{\boldsymbol{v}} - T|^2 + \alpha\,|\nabla \boldsymbol{v}|^2 + \beta\,|\boldsymbol{v} - \tilde{\boldsymbol{u}}|^2 \; d\Omega \right\}, \; \tilde{\boldsymbol{u}}(\boldsymbol{x}) = \boldsymbol{\varphi}^{-1}(\boldsymbol{x}) - \boldsymbol{x}. \end{cases} \tag{5.6}$$

We remark the explicit computation of the inverse displacements in (5.6) is a difficult and computationally expensive task owing to their non-linear nature. However, this kind of model is effective at preventing mesh folding as is illustrated in Figure 5.2 where the mesh problem in Figure 5.2(a) is fixed by the model in Figure 5.2(b). We are motivated to overcome the difficulty of computing the inverse displacements $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ directly, to do this we propose to replace these terms with linear approximations.

This simplification allows us to remove the additional non-linearities from the inverse consistent terms, leaving only the non-linearities seen in diffusion type models, while still retaining the advantages of the inverse consistent model. We know the transformations $\boldsymbol{\varphi}$, $\boldsymbol{\psi}$, and their inverses $\boldsymbol{\varphi}^{-1}$, $\boldsymbol{\psi}^{-1}$, should satisfy the following relations

$$\boldsymbol{\varphi}^{-1}\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) = \boldsymbol{x},\ \boldsymbol{\psi}^{-1}\big(\boldsymbol{\psi}(\boldsymbol{x})\big) = \boldsymbol{x}.$$

Expanding out leads to the following equalities

$$\begin{cases} \boldsymbol{\varphi}^{-1}\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) = \boldsymbol{\varphi}(\boldsymbol{x}) + \tilde{\boldsymbol{u}}\big(\boldsymbol{\varphi}(\boldsymbol{x})\big) = \boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}) + \tilde{\boldsymbol{u}}\big(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})\big) = \boldsymbol{x}, \\ \boldsymbol{\psi}^{-1}\big(\boldsymbol{\psi}(\boldsymbol{x})\big) = \boldsymbol{\psi}(\boldsymbol{x}) + \tilde{\boldsymbol{v}}\big(\boldsymbol{\psi}(\boldsymbol{x})\big) = \boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x}) + \tilde{\boldsymbol{v}}\big(\boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})\big) = \boldsymbol{x} \end{cases}$$

which can be reduced to

$$\begin{cases} \boldsymbol{u}(\boldsymbol{x}) + \tilde{\boldsymbol{u}}\big(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})\big) = 0, \\ \boldsymbol{v}(\boldsymbol{x}) + \tilde{\boldsymbol{v}}\big(\boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})\big) = 0. \end{cases} \tag{5.7}$$

By using a Taylor expansion on the arguments of $\tilde{\boldsymbol{u}}$, $\tilde{\boldsymbol{v}}$ in (5.7), we can obtain the approximations

$$\begin{cases} \tilde{\boldsymbol{u}}\big(\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x})\big) \approx \tilde{\boldsymbol{u}}(\boldsymbol{x}), \\ \tilde{\boldsymbol{v}}\big(\boldsymbol{x} + \boldsymbol{v}(\boldsymbol{x})\big) \approx \tilde{\boldsymbol{v}}(\boldsymbol{x}). \end{cases} \tag{5.8}$$

Substituting (5.8) into (5.7), we get

$$\begin{cases} \boldsymbol{u}(\boldsymbol{x}) \approx -\tilde{\boldsymbol{u}}(\boldsymbol{x}), \\ \boldsymbol{v}(\boldsymbol{x}) \approx -\tilde{\boldsymbol{v}}(\boldsymbol{x}) \end{cases} \tag{5.9}$$

and using (5.9) in (5.5), we have

$$\begin{aligned} \min_{\boldsymbol{u},\boldsymbol{v}} \Big\{ E^{IC}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\int_{\Omega} |T_{\boldsymbol{u}} - R|^2 + |R_{\boldsymbol{v}} - T|^2 + \alpha \left[|\nabla \boldsymbol{u}|^2 + |\nabla \boldsymbol{v}|^2 \right] \\ + \beta \left[|\boldsymbol{u} + \boldsymbol{v}|^2 + |\boldsymbol{v} + \boldsymbol{u}|^2 \right] d\Omega \\ \equiv g^{IC}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \nabla \boldsymbol{u}, \nabla \boldsymbol{v}) \Big\} \end{aligned} \tag{5.10}$$

which results in the following split formulation by alternating minimisation

$$\begin{cases} \min_{\boldsymbol{u}} \left\{ E_1^{IC}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\int_{\Omega} |T_{\boldsymbol{u}} - R|^2 + \alpha\, |\nabla \boldsymbol{u}|^2 + \beta\, |\boldsymbol{u} + \boldsymbol{v}|^2 \, d\Omega \right\}, \\ \\ \min_{\boldsymbol{v}} \left\{ E_2^{IC}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\int_{\Omega} |R_{\boldsymbol{v}} - T|^2 + \alpha\, |\nabla \boldsymbol{v}|^2 + \beta\, |\boldsymbol{v} + \boldsymbol{u}|^2 \, d\Omega \right\}. \end{cases} \tag{5.11}$$

Comparing this model with (5.5), we see the inverse displacements $\tilde{\boldsymbol{u}}$ and $\tilde{\boldsymbol{v}}$ no longer need to be computed directly, instead we need only use the displacements $\boldsymbol{u}$ and $\boldsymbol{v}$.

To solve the minimisation problem (5.11), a discretise-optimise approach (for details see [100, 101]) was used originally. However we instead propose to use an optimise-discretise approach in addition to a fast NMG framework. This approach involves solving the Euler-Lagrange (EL) equations corresponding to (5.11), and can be shown to be given by

$$\begin{cases} -\alpha\Delta u_m + F_m\big(\boldsymbol{u},\boldsymbol{v}\big) = 0, \\ -\alpha\Delta v_m + G_m\big(\boldsymbol{u},\boldsymbol{v}\big) = 0 \end{cases} \tag{5.12}$$

with respective Neumann boundary conditions

$$\nabla u_m \cdot \boldsymbol{n} = 0,\ \nabla v_m \cdot \boldsymbol{n} = 0$$

where $\boldsymbol{n}$ denotes the outward unit normal, and

$$\begin{aligned} F_m\big(\boldsymbol{u},\boldsymbol{v}\big) &= \beta\big[u_m + v_m\big] + \partial_{u_m} T_{\boldsymbol{u}}\big[T_{\boldsymbol{u}} - R\big], \\ G_m\big(\boldsymbol{u},\boldsymbol{v}\big) &= \beta\big[v_m + u_m\big] + \partial_{v_m} R_{\boldsymbol{v}}\big[R_{\boldsymbol{v}} - T\big] \end{aligned} \tag{5.13}$$

denote respectively the force terms for component $m = 1, 2$.

We remark the models in [26, 66, 113, 147], though involving more unknown fields to compute, can also be advantageous when the underlying deformation between T and R is large (and by design the four fields can be small or could be said to be half sized). In this case, it will be of interest to develop fast multigrid methods for them.

5.2.1 Existence of a solution for model (5.10)

Now we prove the existence of solutions for the model (5.10) following the idea of [18] for a similar proof in a related but different model. Given the energy functional $E^{IC}(\boldsymbol{u},\boldsymbol{v})$ defined in (5.10), we wish to show the solutions $\boldsymbol{u}^*$, $\boldsymbol{v}^*$ exist such that $E^{IC}\big(\boldsymbol{u}^*,\boldsymbol{v}^*\big)$ becomes minimal. To do this we use the direct method [40] as in [18], consisting of the following steps:

(i) Take the minimising sequences $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ for E^{IC};

(ii) Show the sequences $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ admit subsequences $\{\boldsymbol{u}_{k_m}, \boldsymbol{v}_{k_m}\}$ which converge to a solution $\big(\boldsymbol{u}^*, \boldsymbol{v}^*\big) \in \chi$ in the weak topology, where χ denotes some function space;

(iii) Show the energy functional E^{IC} is lower semi-continuous.

Before outlining the proof, we first review some necessary theory we use shortly. We begin by introducing three assumptions which will be used throughout the remainder of this proof:

(i) **A1**. Assume $\alpha = \beta = 2$ for simplicity;

(ii) **A2**. Assume the image domain Ω has a C^1 boundary which is denoted by $\partial\Omega$;

(iii) **A3**. Assume R, $T \in C$.

Remark 5.2.1. *Assumptions* **A2** *and* **A3** *are important in order for us to implement the necessary theory for the existence proof. However in practice, images R and T are usually defined in a square domain (with corners) and may not be smooth as functions. We can still deal with this situation. For* **A2**, *we may either smooth the corners of a square domain or consider an enlarged domain which embeds the given square domain. For* **A3**, *if the images R and T are non-smooth, then we would typically use a convolution to produce smooth versions of the original non-smooth images; thus assumption* **A3** *would then be reasonable.*

Second, define the function space χ by the following

$$\chi := W^{1,2}\big(\Omega, \mathbb{R}^2\big) \times W^{1,2}\big(\Omega, \mathbb{R}^2\big)$$

equipped with the norm

$$\big|(\boldsymbol{u}, \boldsymbol{v})\big|_{\chi} = |\boldsymbol{u}|_{W^{1,2}\left(\Omega,\mathbb{R}^2\right)} \times |\boldsymbol{v}|_{W^{1,2}\left(\Omega,\mathbb{R}^2\right)}.$$

Remark 5.2.2. *Here we remark the function space χ is reflexive, this means there exist subsequences which converge in the weak topology. Or, in other words, given the bounded sequences*

$$\big(x_k, y_k\big) \in \chi$$

then there exist subsequences x_{k_m}, y_{k_m} such that

$$\Phi\big(x_{k_m}, y_{k_m}\big) \to \Phi\big(x_k, y_k\big)\, \forall \Phi \in \chi.$$

Third, define the following admissible sets

$$\begin{aligned} \mathcal{A} &= \left\{ \boldsymbol{u} \in \mathcal{A}_0 \colon \left| \int_\Omega \boldsymbol{u}(\boldsymbol{x})\, d\Omega \right| \leq vol(\Omega)\big(M + diam(\Omega)\big) \right\}, \\ \mathcal{B} &= \left\{ \boldsymbol{v} \in \mathcal{B}_0 \colon \left| \int_\Omega \boldsymbol{v}(\boldsymbol{x})\, d\Omega \right| \leq vol(\Omega)\big(N + diam(\Omega)\big) \right\} \end{aligned} \tag{5.14}$$

where

$$\mathcal{A}_0 = \left\{ \boldsymbol{u} \in W^{1,2}\big(\Omega, \mathbb{R}^2\big) \right\}, \mathcal{B}_0 = \left\{ \boldsymbol{v} \in W^{1,2}\big(\Omega, \mathbb{R}^2\big) \right\}$$

and M, $N \in \mathbb{R}$ are some constants.

Definition 5.2.3 (Generalised Poincaré Inequality)**.** *Let $1 \leq p \leq \infty$ and Ω be a bounded connected open subset of $\mathbb{R}^n$ with a Lipschitz boundary, then there exists some constant*

$C \in \mathbb{R}$ which depends only on p and Ω so for every function $\boldsymbol{u} \in W^{1,2}(\Omega)$

$$|\nabla \boldsymbol{u}|_{L^p(\Omega)} \geq C\,|\boldsymbol{u} - \boldsymbol{u}_\Omega|_{L^p(\Omega)}$$

where

$$\boldsymbol{u}_\Omega = \frac{1}{|\Omega|}\int_\Omega \boldsymbol{u}\, d\Omega.$$

Lemma 5.2.4 (General Lower Semi-Continuity)**.** *In the image domain $\Omega \in \mathbb{R}^2$, suppose*

$$f\colon \Omega \to \mathbb{R}^2 \times \mathbb{R}^n \to [0, \infty)$$

is a continuously differentiable function and $f(\cdot, \boldsymbol{y}, \boldsymbol{\xi})$ is measurable for every

$$(\boldsymbol{y}, \boldsymbol{\xi}) \in \mathbb{R}^2 \times \mathbb{R}^n.$$

Also suppose $f(\boldsymbol{x}, \boldsymbol{y}, \cdot)$ is convex and

$$\boldsymbol{y}_k \to \boldsymbol{y} \text{ in } L^p(\Omega, \mathbb{R}^2) \text{ and } \boldsymbol{\xi}_k \to \boldsymbol{\xi} \text{ in } L^p(\Omega, \mathbb{R}^n) \text{ for } p \geq 1.$$

Then the following result holds

$$\lim_{k\to\infty}\left\{\inf\left\{\int_\Omega f(\boldsymbol{x}, \boldsymbol{y}_k(\boldsymbol{x}), \boldsymbol{\xi}_k(\boldsymbol{x}))\, d\Omega\right\}\right\} \geq \int_\Omega f(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}), \boldsymbol{\xi}(\boldsymbol{x}))\, d\Omega$$

Lemma 5.2.5 (Coercity Condition)**.** *Let the assumptions* **A1** *and* **A2** *from earlier hold, then the inverse consistent model* (5.10) *satisfies the coercity condition. In other words, there exist constants $0 < C$, $K \in \mathbb{R}$ such that $\forall\, \boldsymbol{u} \in \mathcal{A}$, $\boldsymbol{v} \in \mathcal{B}$ the following inequality holds*

$$E^{IC}(\boldsymbol{u}, \boldsymbol{v}) \geq K + C\left[|\boldsymbol{u}|^2_{W^{1,2}(\Omega,\mathbb{R}^2)} + |\boldsymbol{v}|^2_{W^{1,2}(\Omega,\mathbb{R}^2)}\right]$$

where $\mathcal{A}$, $\mathcal{B}$ are the admissible sets defined in (5.14)*.*

Proof. Suppose we have some arbitrary transformations $\boldsymbol{u} \in \mathcal{A}$, $\boldsymbol{v} \in \mathcal{B}$, then we have

$$\begin{aligned} E^{IC}(\boldsymbol{u}, \boldsymbol{v}) &= \int_\Omega \frac{1}{2}\left[|T_{\boldsymbol{u}} - R|^2 + |R_{\boldsymbol{v}} - T|^2\right] + |\nabla \boldsymbol{u}|^2 + |\nabla \boldsymbol{v}|^2 \\ &\quad + |\boldsymbol{u} + \boldsymbol{v}|^2 + |\boldsymbol{v} + \boldsymbol{u}|^2\, d\Omega \\ &\geq \int_\Omega |\nabla \boldsymbol{u}|^2 + |\nabla \boldsymbol{v}|^2\, d\Omega \end{aligned} \tag{5.15}$$

since

$$|T_{\boldsymbol{u}} - R|^2 \geq 0,\ |R_{\boldsymbol{v}} - T|^2 \geq 0,\ |\boldsymbol{u} + \boldsymbol{v}|^2 \geq 0,\ |\boldsymbol{v} + \boldsymbol{u}|^2 \geq 0.$$

Then, as a result of assumption **A2**, we can use the generalised Poincaré inequality

(Definition 5.2.3) to get

$$|\nabla \boldsymbol{u}|^2_{L^2} \geq C_1 \, |\boldsymbol{u}|^2_{L^2} - C_1 \, |\Omega| \left[\frac{1}{|\Omega|} \left| \int_\Omega \boldsymbol{u} \, d\Omega \right| \right]^2$$

where $C_1 \in \mathbb{R}$ is some constant. Since we know $\boldsymbol{u} \in \mathcal{A}$ and

$$\left| \int_\Omega \boldsymbol{u} \, d\Omega \right| \leq vol(\Omega)\big(M + diam(\Omega)\big)$$

then we also know there exists some $K_1 \in \mathbb{R}$ such that

$$|\nabla \boldsymbol{u}|^2_{L^2} \geq K_1 + C_1 \, |\boldsymbol{u}|^2_{L^2} \, .$$

Using an analogous argument, and the fact $\boldsymbol{v} \in \mathcal{B}$ with

$$\left| \int_\Omega \boldsymbol{v} \, d\Omega \right| \leq vol(\Omega)\big(N + diam(\Omega)\big)$$

we can show there exist $C_2, K_2 \in \mathbb{R}$ such that the following inequality holds

$$|\nabla \boldsymbol{v}|^2_{L^2} \geq K_2 + C_2 \, |\boldsymbol{v}|^2_{L^2} \, . \tag{5.16}$$

Then introducing the new constants C, $K \in \mathbb{R}$, and combining (5.15)-(5.16), we get

$$E^{IC}(\boldsymbol{u}, \boldsymbol{v}) \geq K + C \left[|\boldsymbol{u}|^2_{W^{1,2}\left(\Omega, \mathbb{R}^2\right)} + |\boldsymbol{v}|^2_{W^{1,2}\left(\Omega, \mathbb{R}^2\right)} \right]$$

and so the coercity condition holds. □

Finally, in order for a solution to the inverse consistent model (5.10) to exist, the following existence theorem must hold

Theorem 5.2.6. *Given the assumptions* **A1-A3** *hold, then the model* (5.10) *with energy functional* $E^{IC}(\boldsymbol{u}, \boldsymbol{v})$ *possesses at least one minimiser* $\big(\boldsymbol{u}^*, \boldsymbol{v}^*\big)$, $\boldsymbol{u}^* \in \mathcal{A}$, $\boldsymbol{v}^* \in \mathcal{B}$.

Proof. We begin by constructing the minimising sequences $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ such that

$$\lim_{k \to \infty} \left\{ E^{IC}\big(\boldsymbol{u}_k, \boldsymbol{v}_k\big) \right\} = \inf_{\boldsymbol{u} \in \mathcal{A}, \boldsymbol{v} \in \mathcal{B}} \left\{ E^{IC}(\boldsymbol{u}, \boldsymbol{v}) \right\}$$

given the energy functional E^{IC} is positive and has a lower bound 0. Moreover, the energy functional $E^{IC}\big(\boldsymbol{x}, \boldsymbol{x}\big)$ is finite. Then, using Lemma 5.2.5, for each n we have

$$M \geq E^{IC}\big(\boldsymbol{u}_k, \boldsymbol{v}_k\big) \geq K + C \left[|\boldsymbol{u}|^2_{W^{1,2}\left(\Omega, \mathbb{R}^2\right)} + |\boldsymbol{v}|^2_{W^{1,2}\left(\Omega, \mathbb{R}^2\right)} \right]$$

and so the sequences $\{\boldsymbol{u}_k, \boldsymbol{v}_k\}$ are bounded in the function space χ. Since we know the function space χ is reflexive (Remark 5.2.2), then this implies there exist some subsequences $\{\boldsymbol{u}_{k_m}, \boldsymbol{v}_{k_m}\}$ which converge to $\big(\boldsymbol{u}^*, \boldsymbol{v}^*\big)$ in the weak topology. Now we see

$$\big(\boldsymbol{u}_{k_m}, \boldsymbol{v}_{k_m}\big) \to \big(\boldsymbol{u}^*, \boldsymbol{v}^*\big) \tag{5.17}$$

in the space $W^{1,2}$ implies (5.17) also holds in the L^2 space owing to the fact $W^{1,2}$ is compactly embedded in the L^2 space i.e. $W^{1,2} \subset\subset L^2$ [45]. From assumption **A2** we know the function g^{IC}, defined in (5.10), is convex for fixed $\boldsymbol{x}$, $\boldsymbol{u}$, $\boldsymbol{v}$, continuously differentiable and measurable in $\boldsymbol{x}$ for fixed

$$\big(\boldsymbol{u}, \boldsymbol{v}, \nabla\boldsymbol{u}, \nabla\boldsymbol{v}\big) \in \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{R}^{2\times 2} \times \mathbb{R}^{2\times 2}.$$

Therefore, using Lemma 5.2.4, we can say the functional E^{IC} is weakly lower semi-continuous. In other words we have

$$\begin{aligned}&\lim_{k\to\infty}\left\{\inf\left\{\int_\Omega g^{IC}\big(\boldsymbol{x}, \boldsymbol{u}_{k_m}, \boldsymbol{v}_{k_m}, \nabla\boldsymbol{u}_{k_m}, \nabla\boldsymbol{v}_{k_m}\big)\, d\Omega\right\}\right\}\\ &\geq \int_\Omega g^{IC}\big(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{v}, \nabla\boldsymbol{u}, \nabla\boldsymbol{v}\big)\, d\Omega.\end{aligned}$$

Thus we have

$$\begin{aligned}\inf_{\boldsymbol{u}\in\mathcal{A},\boldsymbol{v}\in\mathcal{B}}\left\{E^{IC}(\boldsymbol{u},\boldsymbol{v})\right\} &= \lim_{k\to\infty}\left\{E^{IC}\big(\boldsymbol{u}_{k_m}, \boldsymbol{v}_{k_m}\big)\right\}\\ &\geq E^{IC}\big(\boldsymbol{u}^*, \boldsymbol{v}^*\big) \geq \inf_{\boldsymbol{u}\in\mathcal{A},\boldsymbol{v}\in\mathcal{B}}\left\{E^{IC}(\boldsymbol{u},\boldsymbol{v})\right\}.\end{aligned}$$

Therefore, the solution $\big(\boldsymbol{u}^*, \boldsymbol{v}^*\big)$ is a minimiser of the energy functional E^{IC}. □

Remark 5.2.7. *Here we note this proof can also be used to show the existence of solutions for the original Christensen-Johnson model* (5.5) *using a slight modification in* (5.15).

5.2.2 Discretisation of the inverse consistent model (5.12)

To solve the system of EL equations (5.12), we look to obtain a numerical approximation. We do this by discretising the image domain Ω^h into a uniform $n\times n$ mesh, with interval width

$$h = \frac{1}{n-1}$$

and then using a finite difference (FD) method.

Remark 5.2.8. *In general we need not discretise Ω^h using a square mesh, and can instead be discretised using a $n \times m$ mesh where $n \neq m$. However it is common for lung CT slices to be square, and for this reason we work with a square mesh (by taking $m = n$).*

Doing this, in addition to using a lexicographic ordering of the discrete grid points (i, j), we obtain the following discrete versions of (5.12)

$$\begin{cases}-\alpha\big(\Delta^h u_m^h\big)_k + \big(F_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k = 0,\\ -\alpha\big(\Delta^h v_m^h\big)_k + \big(G_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k = 0\end{cases} \tag{5.18}$$

where

$$\left(\Delta^h u_m^h\right)_k \approx \frac{1}{h^2}\Big[\left(u_m^h\right)_{k-n} + \left(u_m^h\right)_{k-1} - 4\left(u_m^h\right)_k + \left(u_m^h\right)_{k+1} + \left(u_m^h\right)_{k+n}\Big]$$

and similar for $\left(\Delta^h v_m^h\right)_k$, also with the following discrete force terms

$$\begin{aligned}\left(F_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\right)_k &= \beta\Big[\left(u_m^h\right)_k + \left(v_m^h\right)_k\Big] + \left(\partial^h_{u_m} T^h_{\boldsymbol{u}}\right)_k\Big[\left(T^h_{\boldsymbol{u}}\right)_k - \left(R^h\right)_k\Big],\\ \left(G_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\right)_k &= \beta\Big[\left(v_m^h\right)_k + \left(u_m^h\right)_k\Big] + \left(\partial^h_{v_m} R^h_{\boldsymbol{v}}\right)_k\Big[\left(R^h_{\boldsymbol{v}}\right)_k - \left(T^h\right)_k\Big]\end{aligned} \tag{5.19}$$

where

$$\begin{aligned}\left(\partial^h_{u_1} T^h_{\boldsymbol{u}}\right)_k &\approx \frac{1}{2h}\Big[\left(T^h_{\boldsymbol{u}}\right)_{k+1} - \left(T^h_{\boldsymbol{u}}\right)_{k-1}\Big],\ \left(\partial^h_{u_2} T^h_{\boldsymbol{u}}\right)_k \approx \frac{1}{2h}\Big[\left(T^h_{\boldsymbol{u}}\right)_{k+n} - \left(T^h_{\boldsymbol{u}}\right)_{k-n}\Big],\\ \left(\partial^h_{v_1} R^h_{\boldsymbol{v}}\right)_k &\approx \frac{1}{2h}\Big[\left(R^h_{\boldsymbol{v}}\right)_{k+1} - \left(R^h_{\boldsymbol{v}}\right)_{k-1}\Big],\ \left(\partial^h_{v_2} R^h_{\boldsymbol{v}}\right)_k \approx \frac{1}{2h}\Big[\left(R^h_{\boldsymbol{v}}\right)_{k+n} - \left(R^h_{\boldsymbol{v}}\right)_{k-n}\Big]\end{aligned}$$

for

$$k = (j-2)(n-1) + (i-1)$$

for $m = 1, 2$ and $i,\, j = 2, \ldots, n-1$. There exits a wide selection of potential methods which we could use to solve the discrete system of equations (5.18). Some examples include the Newton method [19], the time-marching method [47–49, 72, 74, 90, 100] and the additive operator splitting (AOS) method [100, 139]. However for highly non-linear equations, like the ones in (5.18), it can be difficult to ensure these methods converge to a solution. Moreover, for large images, using such methods to solve (5.18) on a single level is extremely expensive computationally. Furthermore owing to the inverse consistent model requiring the simultaneous computation of the forward and backward problems, this expense is doubled. This problem is very common within variational models however, and as such there has been a lot of research into the development of NMG methods with the purpose of greatly reducing CPU cost in solving such problems [33,52,68,72,73,117]. In particular we note the work done by Chumchob and Chen in [33] where they developed a robust NMG framework for diffusion type models (although their model cannot avoid mesh folding).

Now we propose to use a similar NMG framework applied to our inverse consistent model. In addition we also perform a more accurate analysis of the NMG scheme compared with the one presented in [33], in order to obtain a better measure of what is required to achieve optimal convergence for the NMG scheme.

5.2.3 A non-linear multigrid framework

In this subsection we present our NMG framework based upon [33]. Multigrid methods stem from two key observations:

Algorithm 13 $\left[\boldsymbol{u}_h^{(k+1)}, \boldsymbol{v}_h^{(k+1)}\right] \leftarrow ICFASNMG(R^h, T^h, \boldsymbol{n}, \boldsymbol{h}, \boldsymbol{u}_h^{(k)}, \boldsymbol{v}_h^{(k)}, \boldsymbol{\mathcal{G}}_1^h, \boldsymbol{\mathcal{G}}_2^h, \alpha, \beta, \nu_1, \nu_2)$

1: Pre-smoothing step by performing ν_1 steps to update $\boldsymbol{u}_h$, $\boldsymbol{v}_h$

$$\tilde{\boldsymbol{u}}_h^{(k)} \leftarrow Smooth(R^h, T^h, \boldsymbol{u}_h^{(k)}, \boldsymbol{\mathcal{G}}_1^h, \alpha, \beta, \nu_1),\ \tilde{\boldsymbol{v}}_h^{(k)} \leftarrow Smooth(R^h, T^h, \boldsymbol{v}_h^{(k)}, \boldsymbol{\mathcal{G}}_2^h, \alpha, \beta, \nu_1)$$

2: Coarse grid correction

Compute the residuals $\boldsymbol{r}_{1\,h}^{(k)} = \boldsymbol{\mathcal{G}}_1^h - \boldsymbol{\mathcal{N}}_1^h[\boldsymbol{u}_h^{(k)}, \tilde{\boldsymbol{v}}_h^{(k)}]$, $\boldsymbol{r}_{2\,h}^{(k)} = \boldsymbol{\mathcal{G}}_2^h - \boldsymbol{\mathcal{N}}_2^h[\boldsymbol{v}_h^{(k)}, \tilde{\boldsymbol{u}}_h^{(k)}]$

Restrict residuals and smooth approximations $\boldsymbol{r}_{m\,H}^{(k)} = \mathcal{R}_h^H \boldsymbol{r}_{m\,h}^{(k)}$, $\tilde{\boldsymbol{u}}_H^{(k)} = \mathcal{R}_h^H \tilde{\boldsymbol{u}}_h^{(k)}$, $\tilde{\boldsymbol{v}}_H^{(k)} = \mathcal{R}_h^H \tilde{\boldsymbol{v}}_h^{(k)}$

Set $\boldsymbol{H} = 2\boldsymbol{h}$

Form RHS of coarse grid PDEs $\boldsymbol{\mathcal{G}}_1^H = \boldsymbol{r}_1^H + \boldsymbol{\mathcal{N}}_1^H[\tilde{\boldsymbol{u}}_H^{(k)}, \tilde{\boldsymbol{v}}_H^{(k)}]$, $\boldsymbol{\mathcal{G}}_2^H = \boldsymbol{r}_2^H + \boldsymbol{\mathcal{N}}_2^H[\tilde{\boldsymbol{u}}_H^{(k)}, \tilde{\boldsymbol{v}}_H^{(k)}]$

3: Solve to obtain high accuracy solutions $\boldsymbol{u}_H^{(k)}$, $\boldsymbol{v}_H^{(k)}$ using a coarsest grid solver.

Compute the corrections $\boldsymbol{e}_{1\,H}^{(k)} = \boldsymbol{u}_H^{(k)} - \tilde{\boldsymbol{u}}_H^{(k)}$, $\boldsymbol{e}_{2\,H}^{(k)} = \boldsymbol{v}_H^{(k)} - \tilde{\boldsymbol{v}}_H^{(k)}$

Interpolate the corrections to the original fine grid level $\boldsymbol{e}_{1\,h}^{(k)} = \mathcal{I}_H^h \boldsymbol{e}_{1\,H}^{(k)}$, $\boldsymbol{e}_{2\,h}^{(k)} = \mathcal{I}_H^h \boldsymbol{e}_{2\,H}^{(k)}$

Update current grid level approximations using correction $\hat{\boldsymbol{u}}_h^{(k)} = \tilde{\boldsymbol{u}}_h^{(k)} + \boldsymbol{e}_{1\,h}^{(k)}$, $\hat{\boldsymbol{v}}_h^{(k)} = \tilde{\boldsymbol{v}}_h^{(k)} + \boldsymbol{e}_{2\,h}^{(k)}$

4: Post-smoothing step by performing ν_2 steps

$$\boldsymbol{u}_h^{(k+1)} \leftarrow Smooth(R^h, T^h, \hat{\boldsymbol{u}}_h^{(k)}, \boldsymbol{\mathcal{G}}_1^h, \alpha, \beta, \nu_2),\ \boldsymbol{v}_h^{(k+1)} \leftarrow Smooth(R^h, T^h, \hat{\boldsymbol{v}}_h^{(k)}, \boldsymbol{\mathcal{G}}_2^h, \alpha, \beta, \nu_2)$$

(i) Iterative solvers, such as the Gauss-Seidel method, are effective at removing (smoothing) high frequency error components within a small number of iterations. Low frequency error components dominate convergence rates;

(ii) Smooth errors (low frequency) are well approximated on coarser grids. Coarser grids have fewer unknowns making it feasible to do a larger number of iterations without increasing the overall cost.

Using these observations, we can restrict our problem on a fine grid to a much coarser grid, by alternating both smoothing and coarsening steps. On this very coarse grid, we are able to obtain a much more accurate approximation in significantly less time, and from this accurate approximation we can interpolate back up to our original fine grid to obtain an approximation to the original problem. Now we briefly outline our proposed 'full approximation scheme' NMG (FAS-NMG) algorithm (see [9] for details) within the two-grid setting. We begin by denoting the original fine grid by Ω^h and the coarse grid by Ω^H with intervals

$$\boldsymbol{h} = \big(h_1, h_1\big) = \left(\frac{1}{n-1}, \frac{1}{n-1}\right),\ \boldsymbol{H} = 2\boldsymbol{h}$$

respectively. Next we write the PDEs from (5.18) using the following operator notation

$$\begin{cases} \boldsymbol{\mathcal{N}}_1^h[\boldsymbol{u}^h, \boldsymbol{v}^h] = \boldsymbol{\mathcal{G}}_1^h, \\ \boldsymbol{\mathcal{N}}_2^h[\boldsymbol{u}^h, \boldsymbol{v}^h] = \boldsymbol{\mathcal{G}}_2^h \end{cases}$$

where $\boldsymbol{\mathcal{N}}_m^h$ and $\boldsymbol{\mathcal{G}}_m^h$ $(m = 1, 2)$ are sized 2 vectors consisting of the non-linear LHS and initial zero RHS of the discrete EL equations (5.18) for $\boldsymbol{u}^h$, $\boldsymbol{v}^h$ respectively. Then the FAS-NMG framework, in the two-grid setting, is shown in Algorithm 13. This Algorithm 13 can be refined on its coarse grid to recursively interact with increasingly

coarser grids until a desired level is reached (e.g. 8×8), thus leading to the full V-cycle scheme.

Of the three main steps in the NMG framework (smoothing, coarse grid solution, correction), the smoothing step is the most crucial to the convergence of the scheme. As was highlighted by **O2**, only 'smooth' errors can be approximated on a coarser grid, thus any remaining high frequency error components can no longer be removed once the problem has been restricted to a coarser grid (where high frequency error components from the fine grid are not present or visible). This in turn means the NMG will take longer to converge in addition to being less accurate.

5.2.4 Three collective pointwise smoothers for (5.18)

Here we present three different smoother schemes to be used in our NMG scheme.

First pointwise smoother (S1). For our first smoother we consider the simplest type of smoother scheme to solve the system (5.18), namely we use each equation to update each displacement independently. We do this by using the following fixed point iteration scheme

$$\begin{cases} -\alpha\big(\Delta^h u_m^h\big)_k^{(l+1)} + \big(F_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = 0, \\ -\alpha\big(\Delta^h v_m^h\big)_k^{(l+1)} + \big(G_m(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = 0 \end{cases} \tag{5.20}$$

where

$$\begin{aligned} \big(F_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(u_1^h\big)_k^{(l+1)} + \big(v_1^h\big)_k^{(l)}\Big] \\ - \big(\partial_{u_1}^h T_{\boldsymbol{u}}^h\big)_k^{(l)} &\Big[\big(T^h(x_1 + u_1^{(l+1)}, x_2 + u_2^{(l)})\big)_k - \big(R^h\big)_k\Big], \end{aligned}$$

$$\begin{aligned} \big(F_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(u_2^h\big)_k^{(l+1)} + \big(v_2^h\big)_k^{(l)}\Big] \\ -\big(\partial_{u_2}^h T_{\boldsymbol{u}}^h\big)_k^{(l)} &\Big[\big(T^h(x_1 + u_1^{(l)}, x_2 + u_2^{(l+1)})\big)_k - \big(R^h\big)_k\Big], \end{aligned}$$

$$\begin{aligned} big(G_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(v_1^h\big)_k^{(l+1)} + \big(u_1^h\big)_k^{(l)}\Big] \\ -\big(\partial_{v_1}^h R_{\boldsymbol{v}}^h\big)_k^{(l)} &\Big[\big(R^h(x_1 + v_1^{(l+1)}, x_2 + v_2^{(l)})\big)_k - \big(T^h\big)_k\Big], \end{aligned}$$

$$\begin{aligned} \big(G_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(v_2^h\big)_k^{(l+1)} + \big(u_2^h\big)_k^{(l)}\Big] \\ -\big(\partial_{v_2}^h R_{\boldsymbol{v}}^h\big)_k^{(l)} &\Big[\big(R^h(x_1 + v_1^{(l)}, x_2 + v_2^{(l+1)})\big)_k - \big(T^h\big)_k\Big]. \end{aligned} \tag{5.21}$$

Now in order to deal with the non-linearities in the force terms of (5.20), we use the same treatment which was used in [33], namely we linearise the force terms using first

order Taylor expansions. Replacing the non-linear force terms in (5.20), with their first order approximations, leads to the following smoother scheme at step (l) to update the $(l+1)$ terms

$$\begin{cases} -\alpha\big(\Delta^h u_m^h\big)_k^{(l+1)} + \beta\Big[\big(u_m^h\big)_k^{(l+1)} + \big(v_m^h\big)_k^{(l)}\Big] \\ + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_k^{(l)} + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(u_m^h\big)_k^{(l+1)} - \big(u_m^h\big)_k^{(l)}\Big] - \big(R^h\big)_k\Big] = 0, \\ \\ -\alpha\big(\Delta^h v_m^h\big)_k^{(l+1)}\beta\Big[\big(v_m^h\big)_k^{(l+1)} + \big(u_m^h\big)_k^{(l)}\Big] \\ + \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R_{\boldsymbol{v}}^h\big)_k^{(l)} + \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(v_m^h\big)_k^{(l+1)} - \big(v_m^h\big)_k^{(l)}\Big] - \big(T^h\big)_k\Big] = 0 \end{cases} \tag{5.22}$$

where

$$\big(T_{\boldsymbol{u}}^h\big)_k^{(l)} \equiv \big(T^h(x_1 + u_1^{(l)}, x_2 + u_2^{(l)})\big)_k$$

etc. for $m = 1, 2$. In order to compute the $(l+1)$ terms in (5.22), we use a lexicographic Gauss-Seidel (GSLEX) based method.

Second pointwise smoother (S2). Following the smoother scheme proposed by Chumchob and Chen [33], for our second proposed smoother we fully couple all four PDEs together by using a similar scheme to (5.20) and new fixed point linearisations as followed

$$\begin{aligned} \big(F_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(u_1^h\big)_k^{(l+1)} + \big(v_1^h\big)_k^{(l+1)}\Big] - \big(\partial_{u_1}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_k^{(l+1)} - \big(R^h\big)_k\Big], \\ \big(F_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(u_2^h\big)_k^{(l+1)} + \big(v_2^h\big)_k^{(l+1)}\Big] - \big(\partial_{u_2}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_k^{(l+1)} - \big(R^h\big)_k\Big], \end{aligned}$$

$$\begin{aligned} \big(G_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(v_1^h\big)_k^{(l+1)} + \big(u_1^h\big)_k^{(l+1)}\Big] - \big(\partial_{v_1}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R_{\boldsymbol{v}}^h\big)_k^{(l+1)} - \big(T^h\big)_k\Big], \\ \big(G_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} &= \beta\Big[\big(v_2^h\big)_k^{(l+1)} + \big(u_2^h\big)_k^{(l+1)}\Big] - \big(\partial_{v_2}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R_{\boldsymbol{v}}^h\big)_k^{(l+1)} - \big(T^h\big)_k\Big]. \end{aligned} \tag{5.23}$$

Next we linearise the force terms (5.23) by applying Taylor approximations to the discrete force terms (5.23), this results in the following smoother scheme to update the $(l+1)$ terms at step (l)

$$\begin{cases} -\alpha\big(\Delta^h u_m^h\big)_k^{(l+1)} + \beta\left[\big(u_m^h\big)_k^{(l+1)} + \big(v_m^h\big)_k^{(l+1)}\right] \\ +\big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\left[\big(T_{\boldsymbol{u}}^h\big)_k^{(l)} + \sum_{s=1}^{2}\big(\partial_{u_s}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\left[\big(u_s^h\big)_k^{(l+1)} - \big(u_s^h\big)_k^{(l)}\right]\big(R^h\big)_k\right] = 0, \\ \\ -\alpha\big(\Delta^h v_m^h\big)_k^{(l+1)} + \beta\left[\big(v_m^h\big)_k^{(l+1)} + \big(u_m^h\big)_k^{(l+1)}\right] \\ +\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\left[\big(R_{\boldsymbol{v}}^h\big)_k^{(l)} + \sum_{s=1}^{2}\big(\partial_{v_s}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\left[\big(v_s^h\big)_k^{(l+1)} - \big(v_s^h\big)_k^{(l)}\right]\big(T^h\big)_k\right] = 0 \end{cases} \tag{5.24}$$

for $m = 1, 2$. Similar to **S1**, we use a GSLEX based method on (5.24) to update the $(l+1)$ terms.

Third pointwise smoother (S3). The above 4×4 (5.24) system, which must be solved at every discrete interior point in (5.24), is computationally expensive especially in the case of large images. For this reason we propose an alternate, simplified version of **S2** whch still maintains some coupling within the equations. We propose to use a similar scheme to (5.20), except now we have the following force terms with fixed points specified differently

$$\begin{aligned}
&\big(F_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = \beta\Big[\big(u_1^h\big)_k^{(l+1)} + \big(v_1^h\big)_k^{(l+1)}\Big] \\
&- \big(\partial_{u_1}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T^h(x_1 + u_1^{(l+1)}, x_2 + u_2^{(l)})\big)_k - \big(R^h\big)_k\Big], \\
&\big(F_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = \beta\Big[\big(u_2^h\big)_k^{(l+1)} + \big(v_2^h\big)_k^{(l+1)}\Big] \\
&- \big(\partial_{u_2}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T^h(x_1 + u_1^{(l)}, x_2 + u_2^{(l+1)})\big)_k - \big(R^h\big)_k\Big], \\
&\big(G_1(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = \beta\Big[\big(v_1^h\big)_k^{(l+1)} + \big(u_1^h\big)_k^{(l+1)}\Big] \\
&- \big(\partial_{v_1}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R^h(x_1 + v_1^{(l+1)}, x_2 + v_2^{(l)})\big)_k - \big(T^h\big)_k\Big], \\
&\big(G_2(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_k^{(l+1)} = \beta\Big[\big(v_2^h\big)_k^{(l+1)} + \big(u_2^h\big)_k^{(l+1)}\Big] \\
&- \big(\partial_{v_2}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R^h(x_1 + v_1^{(l)}, x_2 + v_2^{(l+1)})\big)_k - \big(T^h\big)_k\Big]. \qquad (5.25)
\end{aligned}$$

Again, after using Taylor approximations to linearise (5.25), at iteration step (l) we have the following smoother scheme which we use to compute the $(l+1)$ updates

$$\begin{cases}
-\alpha\big(\Delta^h u_m^h\big)_k^{(l+1)} + \beta\Big[\big(u_m^h\big)_k^{(l+1)} + \big(v_m^h\big)_k^{(l+1)}\Big] \\
+\big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_k^{(l)} + \Big[\big(u_m^h\big)_k^{(l+1)} - \big(u_m^h\big)_k^{(l)}\Big]\big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_k^{(l)} - \big(R^h\big)_k\Big] = 0, \\
\\
-\alpha\big(\Delta^h v_m^h\big)_k^{(l+1)}\beta\Big[\big(v_m^h\big)_k^{(l+1)} + \big(u_m^h\big)_k^{(l+1)}\Big] \\
+\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_k^{(l)}\Big[\big(R_{\boldsymbol{v}}^h\big)_k^{(l)} + \Big[\big(v_m^h\big)_k^{(l+1)} - \big(v_m^h\big)_k^{(l)}\Big]\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_k^{(l)} - \big(T^h\big)_k\Big] = 0
\end{cases} \qquad (5.26)$$

for $m = 1, 2$. As we did for **S1** and **S2**, we use a scheme based upon a GSLEX method to compute the $(l+1)$ updates in (5.26).

5.3 Analysis for the NMG algorithm

As we mentioned at the end of §5.2.3, the effectiveness of the smoother scheme has a severe impact on the convergence of the NMG scheme. In order to determine how effective a given smoother scheme is within the NMG framework, we look to compute the so called 'smoothing rate' of the scheme which gives us an insight into how effectively the chosen smoother removes high frequency error components. However, before we look at computing the smoothing rates of our three proposed smoothers from §5.2.4, we must first determine whether each of the proposed smoothers are suitable for use as pointwise error smoothing procedures. To do this we must compute the h-ellipticity for each of the proposed smoothers. For both calculations (i.e. smoothing rates and h-ellipticity values) we can use local Fourier analysis or LFA.

5.3.1 Local Fourier analysis (LFA)

In order to analyse the h-ellipticity and smoothing rate of a given smoother scheme, we can use a technique called LFA. Originally LFA was designed to only analyse the smoothing properties of discrete linear operators, however the work done by A. Brandt [9] proposed to locally 'freeze' the coefficients of non-linear operators thus enabling the use of LFA for non-linear operators such as those in (5.20). In LFA [25,33], we begin by considering our problem over an infinite grid (thus removing any influence from the boundary conditions), and then assuming the discrete form of a variable non-linear operator can be replaced locally by a constant linear operator and extended to this infinite grid, which we define as followed

$$\Omega_h^{\infty} := \left\{ \boldsymbol{x} \in \Omega \colon \boldsymbol{x} = \big(x_1, x_2\big)^T = \big(ih, jh\big)^T \text{ for } i,\, j \in \mathbb{Z}^+ \right\} \tag{5.27}$$

with grid interval $\boldsymbol{h}$ defined by

$$\boldsymbol{h} = (h, h) = \left(\frac{1}{n-1}, \frac{1}{n-1} \right).$$

In addition let us also define the grid functions

$$\boldsymbol{\Phi}^h\big(\boldsymbol{x}, \boldsymbol{\theta}\big) = \exp\left(\frac{\boldsymbol{i}\boldsymbol{\theta} \cdot \boldsymbol{x}}{\boldsymbol{h}} \right)$$

where

$$\boldsymbol{\theta} = \big(\theta_1, \theta_2\big)^T \in \boldsymbol{\Theta} = [-\pi, \pi)^2,\ \boldsymbol{x} \in \Omega_h^{\infty},\ \boldsymbol{i} = \sqrt{-1}.$$

Moreover, when we apply a discrete linear operator $\boldsymbol{\mathcal{L}}^h$ to the grid functions $\boldsymbol{\Phi}^h\big(\boldsymbol{x},\boldsymbol{\theta}\big)$, we get the following

$$\boldsymbol{\mathcal{L}}^h\boldsymbol{\Phi}^h\big(\boldsymbol{x},\boldsymbol{\theta}\big) = \hat{\boldsymbol{\mathcal{L}}}^h\big(\boldsymbol{\theta}\big)\boldsymbol{\Phi}^h\big(\boldsymbol{x},\boldsymbol{\theta}\big)$$

where $\hat{\boldsymbol{\mathcal{L}}}^h\big(\boldsymbol{\theta}\big)$ denotes the Fourier symbol of the linear operator $\boldsymbol{\mathcal{L}}^h$ (see [131,141]).

5.3.2 H-ellipticity measure for the proposed smoothers

For effective smoother schemes, the measure of the h-ellipticity is a key component which must be considered. This measure is used to ascertain whether a given smoother scheme, such as those we outlined in §5.2.4, are sufficient for use as pointwise error smoothing procedures for the given discrete operator within a multigrid framework. If not, one must consider line or block smoothers or a reformulation of the problem.

We now demonstrate our proposed smoothers from §5.2.4 can be constructed for the given discrete operator, and can therefore be used in our proposed NMG scheme. To do this we use a similar calculation to those shown in [33,68,86,131,141] applied to the smoother schemes (5.22), (5.24) and (5.26) at some given outer iteration step.

H-ellipticity for smoother S1. We begin by writing (5.22) in the following operator form

$$\boldsymbol{\mathcal{L}}_1^h\boldsymbol{w}^h = \boldsymbol{\mathcal{G}}^h \tag{5.28}$$

with

$$\begin{aligned}
\boldsymbol{\mathcal{L}}_1^h &= \begin{bmatrix} -\alpha\Delta^h + \sigma_{11}^h + \beta & 0 & 0 & 0 \\ 0 & -\alpha\Delta^h + \sigma_{22}^h + \beta & 0 & 0 \\ 0 & 0 & -\alpha\Delta^h + \tau_{11}^h + \beta & 0 \\ 0 & 0 & 0 & -\alpha\Delta^h + \tau_{22}^h + \beta \end{bmatrix}, \\
\boldsymbol{\mathcal{G}}^h &= \begin{bmatrix} \boldsymbol{g}_1^h - F_1\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) \\ \boldsymbol{g}_2^h - F_2\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) \\ \boldsymbol{g}_3^h - G_1\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) \\ \boldsymbol{g}_4^h - G_2\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) \end{bmatrix}, \ \boldsymbol{w}^h = \begin{bmatrix} \boldsymbol{u}_1^h \\ \boldsymbol{u}_2^h \\ \boldsymbol{v}_1^h \\ \boldsymbol{v}_2^h \end{bmatrix}
\end{aligned} \tag{5.29}$$

and where

$$\begin{aligned}
F_m\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) &= \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)^2 u_m^h - \beta v_m^h - \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)\big[T_{\boldsymbol{u}}^h - R^h\big], \\
G_m\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) &= \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)^2 v_m^h - \beta u_m^h - \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)\big[R_{\boldsymbol{v}}^h - T^h\big], \\
\sigma_{pq}^h &= \big(\partial_{u_p}^h T_{\boldsymbol{u}}^h\big)\big(\partial_{u_q}^h T_{\boldsymbol{u}}^h\big), \ \tau_{pq}^h = \big(\partial_{v_p}^h R_{\boldsymbol{v}}^h\big)\big(\partial_{v_q}^h R_{\boldsymbol{v}}^h\big)
\end{aligned} \tag{5.30}$$

for m, p, $q = 1, 2$. Since LFA is a local method for a non-linear problem, we apply the analysis separately to each individual grid point. This then leads to a local discrete

system which is only defined within a small neighbourhood of the discrete grid point (i,j). Applying our discrete linear operator $\boldsymbol{\mathcal{L}}_1^h$ to the grid functions $\boldsymbol{\Phi}^h(\boldsymbol{x},\boldsymbol{\theta})$ yields the following

$$\boldsymbol{\mathcal{L}}_1^h\boldsymbol{\Phi}^h(\boldsymbol{x},\boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\boldsymbol{\Phi}^h(\boldsymbol{x},\boldsymbol{\theta})$$

where $\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the operator $\boldsymbol{\mathcal{L}}_1^h$, and is given by

$$\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h + a^h & 0 & 0 & 0 \\ 0 & \sigma_{22}^h + a^h & 0 & 0 \\ 0 & 0 & \tau_{11}^h + a^h & 0 \\ 0 & 0 & 0 & \tau_{22}^h + a^h \end{bmatrix}$$

with

$$a^h = \beta - \alpha\hat{\mathscr{L}}^h(\boldsymbol{\theta}) \tag{5.31}$$

and $\hat{\mathscr{L}}^h(\boldsymbol{\theta})$ denoting the Fourier symbol of the discrete Laplace operator Δ^h. Then, the h-ellipticity measure is calculated from the following

$$\mathscr{E}_1^h(\boldsymbol{\mathcal{L}}_1^h) = \frac{\min\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta}\in\boldsymbol{\Theta}_{high}\right\}}{\max\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta}\in\boldsymbol{\Theta}\right\}} \tag{5.32}$$

where

$$\boldsymbol{\Theta} = [-\pi,\pi)^2,\ \boldsymbol{\Theta}_{high} = \boldsymbol{\Theta}\backslash\left[-\frac{\pi}{2},\frac{\pi}{2}\right)^2.$$

It can be shown

$$\begin{aligned}\det\left(\hat{\boldsymbol{\mathcal{L}}}^h(\boldsymbol{\theta})\right) = {}& \alpha^4\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^4 - \alpha^3(d_1+c_1)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^3 \\ & +\alpha^2(d_2+c_1d_1+c_2)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^2 - \alpha(c_1d_2+c_2d_1)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big) + c_2d_2\end{aligned}$$

where

$$\begin{aligned} &c_1 = \sigma_{11}^h + \sigma_{22}^h + 2\beta,\ c_2 = \sigma_{11}^h\sigma_{22}^h + \beta\big(\sigma_{11}^h + \sigma_{22}^h\big) + \beta^2, \\ &d_1 = \tau_{11}^h + \tau_{22}^h + 2\beta,\ d_2 = \tau_{11}^h\tau_{22}^h + \beta\big(\tau_{11}^h + \tau_{22}^h\big) + \beta^2. \end{aligned} \tag{5.33}$$

From [33], it was shown

$$\begin{aligned} -\hat{\mathscr{L}}^h(\boldsymbol{\theta}) &= \frac{2}{h^2}\Big[2 - \big(\cos\theta_1 + \cos\theta_2\big)\Big], \\ \min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{high}}\Big\{\big(-\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)\Big\} &= \frac{2}{h^2},\ \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\Big\{\big(-\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)\Big\} = \frac{8}{h^2}. \end{aligned} \tag{5.34}$$

Thus (5.32) becomes

$$\mathscr{E}_1^h(\boldsymbol{\mathcal{L}}_1^h) = \frac{\left[\frac{16\alpha^4}{h^8} + \frac{8\alpha^3\left(d_1+c_1\right)}{h^6} + \frac{4\alpha^2\left(d_+c_1d_1+c_2\right)}{h^4} + \frac{2\alpha\left(c_1d_2+c_2d_1\right)}{h^2} + c_2d_2\right]}{\left[\frac{4096\alpha^4}{h^8} + \frac{512\alpha^3\left(d_1+c_1\right)}{h^6} + \frac{64\alpha^2\left(d_+c_1d_1+c_2\right)}{h^4} + \frac{8\alpha\left(c_1d_2+c_2d_1\right)}{h^2} + c_2d_2\right]}$$

$$= \frac{\left[\begin{array}{r}16\alpha^4 + 8\alpha^3\left(d_1+c_1\right)h^2 + 4\alpha^2\left(d_+c_1d_1+c_2\right)h^4 \\ + 2\alpha\left(c_1d_2+c_2d_1\right)h^6 + c_2d_2h^8\end{array}\right]}{\left[\begin{array}{r}4096\alpha^4 + 512\alpha^3\left(d_1+c_1\right)h^2 + 64\alpha^2\left(d_+c_1d_1+c_2\right)h^4 \\ + 8\alpha\left(c_1d_2+c_2d_1\right)h^6 + \left(c_2d_2\right)h^8\end{array}\right]}$$

and so, taking the limit as $h \to 0$, we get

$$\lim_{h\to 0}\left\{\mathscr{E}_1^h(\boldsymbol{\mathcal{L}}_1^h)\right\} = \frac{1}{256}. \tag{5.35}$$

From this result, we can conclude the h-ellipticity measure is always bounded away from 0 regardless of the values of α, β, h, σ_{pq}^h, τ_{pq}^h for p, $q = 1, 2$. Or in other words, the results do not depend on the given images R, T, the choice of parameters α, β or the mesh interval h. Therefore we conclude smoother **S1** is sufficient for use as a pointwise error smoothing procedure.

H-ellipticity for smoother S2. Now we repeat the h-ellipticity calculation procedure for smoother **S2**.

Similar to smoother **S1**, we get the following Fourier symbol for the operator $\boldsymbol{\mathcal{L}}_2^h$

$$\hat{\boldsymbol{\mathcal{L}}}_2^h(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h + a^h & \sigma_{12}^h & \beta & 0 \\ \sigma_{12}^h & \sigma_{22}^h + a^h & 0 & \beta \\ \beta & 0 & \tau_{11}^h + a^h & \tau_{12}^h \\ 0 & \beta & \tau_{12}^h & \tau_{22}^h + a^h \end{bmatrix}$$

where a^h is as defined in (5.31), $\mathscr{L}^h(\boldsymbol{\theta})$ again denotes the Fourier symbol of Δ^h and $\sigma_{pq}^h, \tau_{pq}^h$ are as in (5.30). Then we compute the h-ellipticity for $\boldsymbol{\mathcal{L}}_2^h$ using

$$\mathscr{E}_2^h(\boldsymbol{\mathcal{L}}_2^h) = \frac{\min\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_2^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}}{\max\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_2^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}}. \tag{5.36}$$

Simplifying the determinant we get

$$\begin{aligned}\det\big(\hat{\boldsymbol{\mathcal{L}}}_2^h(\boldsymbol{\theta})\big) &= \big(\sigma_{11}^h + a\big)\big(\sigma_{22}^h + a\big)\big(\tau_{11}^h + a\big)\big(\tau_{22}^h + a\big) \\ &\quad - \big(\sigma_{11}^h + a\big)\big(\sigma_{22}^h + a\big)\big(\tau_{12}^h\big)^2 - \big(\tau_{11}^h + a\big)\big(\tau_{22}^h + a\big)\big(\sigma_{12}^h\big)^2 \\ &\quad - \big(\sigma_{11}^h + a\big)\big(\tau_{22}^h + a\big)\beta^2 - \big(\sigma_{22}^h + a\big)\big(\tau_{22}^h + a\big)\beta^2 \\ &\quad + \big(\sigma_{12}^h\big)^2\big(\tau_{12}^h\big)^2 - 2\sigma_{12}^h\tau_{12}^h\beta^2 + \beta^4 \\ \\ &= \alpha^4\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^4 - \alpha^3\big(d_1 + c_1\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^3 \\ &\quad + \alpha^2\big(d_2 + c_1d_1 + c_2 - c_5 - d_5 + 2\beta^2\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^2 \\ &\quad - \alpha\big(c_1d_2 + c_2d_1 + c_3 + d_3 + c_1d_5 + d_1c_5\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big) \\ &\quad + c_2d_2 + c_4 + d_5 - d_2c_5 - c_2d_5 + c_5d_5 + 2\beta^4\end{aligned}$$

where c_1, c_2, d_1, d_2 are as in (5.33), and

$$\begin{aligned}&c_3 = \beta^2\left[\sigma_{11}^h + \tau_{11}^h + 2\beta\right],\ c_4 = \beta^2\left[\beta^2 + \beta\big(\sigma_{11}^h + \tau_{11}^h\big) + \sigma_{11}^h + \tau_{11}^h\right],\ c_5 = \big(\sigma_{12}^h\big)^2, \\ &d_3 = \beta^2\left[\sigma_{22}^h + \tau_{22}^h + 2\beta\right],\ d_4 = \beta^2\left[\beta^2 + \beta\big(\sigma_{22}^h + \tau_{22}^h\big) + \sigma_{22}^h + \tau_{22}^h\right],\ d_5 = \big(\tau_{12}^h\big)^2. \qquad (5.37)\end{aligned}$$

From the h-ellipticity calculation of smoother **S1**, we see the value of the limit (5.35) as $h \to 0$ depends only on the coefficient of the α^4 term. Using this fact we obtain the following

$$\lim_{h\to 0}\left\{\mathscr{E}_2^h\big(\boldsymbol{\mathcal{L}}_2^h\big)\right\} = \frac{1}{256}$$

and so smoother **S2** is also suitable for use as a pointwise error smoothing procedure.

H-ellipticity for smoother S3: Finally we once again repeat the h-ellipticity calculation, this time for our simplified smoother **S3**. Doing so gives the following Fourier symbol corresponding to the operator $\boldsymbol{\mathcal{L}}_3^h$

$$\hat{\boldsymbol{\mathcal{L}}}_3^h(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h + a^h & 0 & \beta & 0 \\ 0 & \sigma_{22}^h + a^h & 0 & \beta \\ \beta & 0 & \tau_{11}^h + a^h & 0 \\ 0 & \beta & 0 & \tau_{22}^h + a^h \end{bmatrix}$$

where a^h is as defined in (5.31), $\hat{\mathscr{L}}^h(\boldsymbol{\theta})$ again denotes the Fourier symbol of the discrete Laplace operator Δ^h and σ_{pq}^h, τ_{pq}^h are as defined in (5.30) for $p, q = 1, 2$. We compute the h-ellipticity using the following

$$\mathscr{E}_3^h\big(\boldsymbol{\mathcal{L}}_3^h\big) = \frac{\min\left\{\left|\det\big(\hat{\boldsymbol{\mathcal{L}}}_3^h(\boldsymbol{\theta})\big)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}}{\max\left\{\left|\det\big(\hat{\boldsymbol{\mathcal{L}}}_3^h(\boldsymbol{\theta})\big)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}}. \qquad (5.38)$$

Again we can simplify the determinant in the following way

$$\begin{aligned}\det\big(\hat{\boldsymbol{\mathcal{L}}}_3^h(\boldsymbol{\theta})\big) &= \big(\sigma_{11}^h + a\big)\big(\sigma_{22}^h + a\big)\big(\tau_{11}^h + a\big)\big(\tau_{22}^h + a\big) \\ &\quad - \big(\sigma_{11}^h + a\big)\big(\tau_{11}^h + a\big)\beta^2 - \big(\sigma_{22}^h + a\big)\big(\tau_{22}^h + a\big)\beta^2 + \beta^4 \\ \\ &= \alpha^4\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^4 - \alpha^3\big(d_1 + c_1\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^3 \\ &\quad + \alpha^2\big(d_2 + c_1 d_1 + c_2 + 2\beta^2\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big)^2 \\ &\quad - \alpha\big(c_1 d_2 + c_2 d_1 + c_3 + d_3\big)\big(\hat{\mathscr{L}}^h(\boldsymbol{\theta})\big) + c_2 d_2 + c_4 + d_4 + \beta^4\end{aligned}$$

where c_1, c_2, d_1, d_2 are as given in (5.33) and c_3, c_4, $_3$, d_4 are as given in (5.37). Then again after taking the limit as $h \to 0$ of (5.38), we get the following

$$\lim_{h\to 0}\Big\{\mathscr{E}_3^h\big(\boldsymbol{\mathcal{L}}_3^h\big)\Big\} = \frac{1}{256}.$$

Thus we reach the same conclusion we did for the previous smoothers, namely the h-ellipticity is always bounded away from 0, and so smoother **S3** is sufficient for use as a pointwise error smoothing procedure.

5.3.3 Smoother analysis of the proposed smoothers

We now consider how effective our smoother schemes from §5.2.4 are at removing high frequency error components. The discrete residual error, as shown in §5.2.3, can be split into the sum of low frequency error components (which can be well approximated on a coarser grid) and high frequency error components (which disappear on coarser grids due to aliasing). For this reason, one key aspect of the NMG framework is the removal of all high frequency error components before we restrict to a coarser grid. We use LFA to approximate the smoothing rate of a given smoother scheme, and from this we can obtain an estimate of how many smoothing steps we need to remove the high frequency components if we aim to reduce the error by 10^{-1} (typical in a NMG context).

LFA for smoother S1. We begin our calculation of the smoothing rate by writing the discrete system (5.22) in the following way

$$\boldsymbol{\mathcal{L}}_1^h \boldsymbol{w}^h + \boldsymbol{\mathcal{M}}_1^h \boldsymbol{w}^h = \boldsymbol{\mathcal{G}}^h \tag{5.39}$$

where $\boldsymbol{\mathcal{L}}_1^h$, $\boldsymbol{w}^h$, $\boldsymbol{\mathcal{G}}^h$ are as defined in (5.29), and

$$\boldsymbol{\mathcal{M}}_1^h = \begin{bmatrix} -\sigma_{11}^h & 0 & \beta & 0 \\ 0 & -\sigma_{22}^h & 0 & \beta \\ \beta & 0 & -\tau_{11}^h & 0 \\ 0 & \beta & 0 & -\tau_{22}^h \end{bmatrix} \tag{5.40}$$

with σ^h_{pq}, τ^h_{pq} as in (5.30) for p, $q = 1, 2$. In addition we can re-write the discrete Laplace operator as

$$\Delta^h = \mathscr{L}^h_+ + \mathscr{L}^h_-$$

where $\mathscr{L}^h_+$, $\mathscr{L}^h_-$ define the following stencils

$$\mathscr{L}^h_+ = \frac{1}{h^2}\begin{bmatrix} 0 & 0 & 0 \\ 1 & -4 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; \mathscr{L}^h_- = \frac{1}{h^2}\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.41}$$

and so, we can write (5.39) in the following way

$$\boldsymbol{\mathcal{L}}^h_{1+}\boldsymbol{u}^h_{new} + \boldsymbol{\mathcal{L}}^h_{1-}\boldsymbol{u}^h_{old} + \boldsymbol{\mathcal{M}}^h_1\boldsymbol{u}^h_{old} = \boldsymbol{\mathcal{G}}^h \tag{5.42}$$

where we have denoted the current and previous approximations of $\boldsymbol{u}^h$, $\boldsymbol{v}^h$ by $\boldsymbol{u}^h_{new}$, $\boldsymbol{v}^h_{new}$ and $\boldsymbol{u}^h_{old}$, $\boldsymbol{v}^h_{old}$ respectively, also with

$$\boldsymbol{\mathcal{L}}^h_{1+} = \begin{bmatrix} a^h + \sigma^h_{11} & 0 & 0 & 0 \\ 0 & a^h + \sigma^h_{22} & 0 & 0 \\ 0 & 0 & a^h + \tau^h_{11} & 0 \\ 0 & 0 & 0 & a^h + \tau^h_{22} \end{bmatrix},$$

$$\boldsymbol{\mathcal{L}}^h_{1-} = \begin{bmatrix} -\alpha\mathscr{L}^h_- & 0 & 0 & 0 \\ 0 & -\alpha\mathscr{L}^h_- & 0 & 0 \\ 0 & 0 & -\alpha\mathscr{L}^h_- & 0 \\ 0 & 0 & 0 & -\alpha\mathscr{L}^h_- \end{bmatrix} \tag{5.43}$$

also

$$a^h = \beta - \alpha\mathscr{L}^h_+$$

and where $\boldsymbol{\mathcal{M}}^h_1$ is as given in (5.40). Now subtracting (5.42) from (5.39) we can obtain the local error equations given by

$$\left[\boldsymbol{\mathcal{L}}^h_{1+}\right]\boldsymbol{e}^h_{new} = -\left[\boldsymbol{\mathcal{L}}^h_{1-} + \boldsymbol{\mathcal{M}}^h_1\right]\boldsymbol{e}^h_{old} \tag{5.44}$$

where

$$\boldsymbol{e}^h_* = \left[e^h_{1*}, e^h_{2*}, e^h_{3*}, e^h_{4*}\right]^T.$$

Then we expand the local errors in (5.44) using Fourier components to give

$$\boldsymbol{e}^h_* = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \boldsymbol{\psi}^*_{\boldsymbol{\theta}} \exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j\right) \tag{5.45}$$

where $\boldsymbol{\psi}^*_{\boldsymbol{\theta}}$ are Fourier coefficients and

$$\boldsymbol{i} = \sqrt{-1},\ \boldsymbol{\Theta} = [-\pi, \pi)^2,\ \lambda_m = \frac{2\theta_m \pi}{h}$$

for $m = 1, 2$. Using the Fourier component form of the errors in (5.45), we can re-write the local error equation (5.44) in terms of these Fourier components. Doing so gives us the following

$$\left[\hat{\boldsymbol{\mathcal{L}}}^h_{1+}(\boldsymbol{\theta})\right] \boldsymbol{\psi}^{new}_{\boldsymbol{\theta}} \exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j\right) = -\left[\hat{\boldsymbol{\mathcal{L}}}^h_{1-}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}^h_1(\boldsymbol{\theta})\right] \boldsymbol{\psi}^{old}_{\boldsymbol{\theta}} \exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j\right)$$

where

$$\begin{aligned}
\hat{\boldsymbol{\mathcal{L}}}^h_{1+}(\boldsymbol{\theta}) &= \begin{bmatrix} \tilde{a}^h_+ + \sigma^h_{11} & 0 & 0 & 0 \\ 0 & \tilde{a}^h_+ + \sigma^h_{22} & 0 & 0 \\ 0 & 0 & \tilde{a}^h_+ + \tau^h_{11} & 0 \\ 0 & 0 & 0 & \tilde{a}^h_+ + \tau^h_{22} \end{bmatrix} \\
\hat{\boldsymbol{\mathcal{L}}}^h_{1-}(\boldsymbol{\theta}) &= \begin{bmatrix} \tilde{a}^h_- & 0 & 0 & 0 \\ 0 & \tilde{a}^h_- & 0 & 0 \\ 0 & 0 & \tilde{a}^h_- & 0 \\ 0 & 0 & 0 & \tilde{a}^h_- \end{bmatrix},\ \hat{\boldsymbol{\mathcal{M}}}^h_1(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma^h_{11} & 0 & \beta & 0 \\ 0 & -\sigma^h_{22} & 0 & \beta \\ \beta & 0 & -\tau^h_{11} & 0 \\ 0 & \beta & 0 & -\tau^h_{22} \end{bmatrix}
\end{aligned} \tag{5.46}$$

with

$$\tilde{a}^h_+ = \beta + \frac{4\alpha}{h^2} - \frac{\alpha}{h^2}\left(e^{-\boldsymbol{i}\lambda_2} + e^{-\boldsymbol{i}\lambda_1}\right),\ \tilde{a}^h_- = \beta + \frac{4\alpha}{h^2} - \frac{\alpha}{h^2}\left(e^{\boldsymbol{i}\lambda_2} + e^{\boldsymbol{i}\lambda_1}\right),\ \lambda_m = \frac{2\pi\theta_m}{h}$$

for $m = 1, 2$. Finally, we compute the local smoothing rate using the following

$$\mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\rho(\hat{\boldsymbol{\mathcal{S}}}^h_1(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\} \tag{5.47}$$

where

$$\boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \backslash \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^2$$

denotes the high frequency range, $\rho(\,\cdot\,)$ the spectral radius and $\hat{\boldsymbol{\mathcal{S}}}^h_1(\boldsymbol{\theta})$ the amplification matrix given by the following

$$\hat{\boldsymbol{\mathcal{S}}}^h_1(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}^h_{1+}(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}^h_{1-}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}^h_1(\boldsymbol{\theta})\right].$$

LFA for smoother S2. Now we repeat the smoothing rate calculation we used for smoother **S1**, but this time for smoother **S2**. Doing so we compute the local smoothing rate from

$$\mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\rho(\hat{\boldsymbol{\mathcal{S}}}^h_2(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}$$

with amplification matrix

$$\hat{\boldsymbol{\mathcal{S}}}_2^h(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{2+}^h(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}_{2-}^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_2^h(\boldsymbol{\theta})\right]$$

where $\hat{\boldsymbol{\mathcal{L}}}_{2-}^h(\boldsymbol{\theta})$ is the same as $\hat{\boldsymbol{\mathcal{L}}}_{1-}^h(\boldsymbol{\theta})$ from (5.46), and

$$\hat{\boldsymbol{\mathcal{L}}}_{2+}^h(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}_+^h + \sigma_{11}^h & \sigma_{12}^h & \beta & 0 \\ \sigma_{12}^h & \tilde{a}_+^h + \sigma_{22}^h & 0 & \beta \\ \beta & 0 & \tilde{a}_+^h + \tau_{11}^h & \tau_{12}^h \\ 0 & \beta & \tau_{12}^h & \tilde{a}_+^h + \tau_{22}^h \end{bmatrix},$$

$$\hat{\boldsymbol{\mathcal{M}}}_2^h(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma_{11}^h & -\sigma_{12}^h & 0 & 0 \\ -\sigma_{12}^h & -\sigma_{22}^h & 0 & 0 \\ 0 & 0 & -\tau_{11}^h & -\tau_{12}^h \\ 0 & 0 & -\tau_{12}^h & -\tau_{22}^h \end{bmatrix}$$

Remark 5.3.1. *We remark if we set $\beta = 0$, then the smoother analysis becomes similar to the one shown in [33]. However the analysis in [33] led to an over-estimation of the smoothing rate due to omitting the lagged displacements (as shown by the $\hat{\boldsymbol{\mathcal{M}}}_2^h(\boldsymbol{\theta})$ matrix), which resulted in an under-estimation of the number of smoother steps required and thus a less effective NMG scheme.*

LFA for smoother S3. Finally we repeat the smoothing rate calculation, this time for smoother **S3**. We compute the local smoothing rate using the following

$$\mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\rho(\hat{\boldsymbol{\mathcal{S}}}_3^h(\boldsymbol{\theta})) : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}$$

with amplification matrix

$$\hat{\boldsymbol{\mathcal{S}}}_3^h(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{3+}^h(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}_{3-}^h(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_3^h(\boldsymbol{\theta})\right]$$

where $\hat{\boldsymbol{\mathcal{L}}}_{3-}^h(\boldsymbol{\theta})$ is the same as $\hat{\boldsymbol{\mathcal{L}}}_{1-}^h(\boldsymbol{\theta})$ from (5.46), and

$$\hat{\boldsymbol{\mathcal{L}}}_{3+}^h(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}_+^h + \sigma_{11}^h & 0 & \beta & 0 \\ 0 & \tilde{a}_+^h + \sigma_{22}^h & 0 & \beta \\ \beta & 0 & \tilde{a}_+^h + \tau_{11}^h & 0 \\ 0 & \beta & 0 & \tilde{a}_+^h + \tau_{22}^h \end{bmatrix},$$

$$\hat{\boldsymbol{\mathcal{M}}}_3^h(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma_{11}^h & 0 & 0 & 0 \\ 0 & -\sigma_{22}^h & 0 & 0 \\ 0 & 0 & -\tau_{11}^h & 0 \\ 0 & 0 & 0 & -\tau_{22}^h \end{bmatrix}.$$

Smoothing rate calculations. From Table 5.1 we see as the value of β increases, the smoothing rate for smoother **S1** gets closer to 1. For this reason we conclude smoother

		S1		S2		S3	
$\boldsymbol{\alpha}$	$\boldsymbol{\beta}$	$\boldsymbol{\mu_{avg}}$	$\boldsymbol{Tol\ 10^{-1}}$	$\boldsymbol{\mu_{avg}}$	$\boldsymbol{Tol\ 10^{-1}}$	$\boldsymbol{\mu_{avg}}$	$\boldsymbol{Tol\ 10^{-1}}$
$\frac{1}{15}$	0	0.72942	8	0.73352	8	0.72942	8
$\frac{1}{15}$	10^2	0.79205	10	0.72972	8	0.72526	8
$\frac{1}{15}$	10^4	0.93335	34	0.73178	8	0.72545	8

Table 5.1: Comparison of the smoothing rates of the proposed smoothers **S1**–**S3** for parameters $\alpha = \frac{1}{15}$ and $\beta = 0$, 10^2, 10^4 after five inner and outer iterations on a 32×32 grid for Example 2 as shown in Figure 5.3. For each smoother, the smoothing rates and number of inner iterations required to reach an error reduction of 10^{-1} are shown.

S1 is not suitable for use in the NMG framework, since this increase in smoothing rate would require an unreasonable number of smoother steps for practical applications as shown by the number of iterations required to reduce the error to a tolerance of 10^{-1} from Table 5.1. We also see the rates for smoothers **S2** and **S3** remain stable even as the value of β increases, and owing to this stability, we see for both smoothers **S2** and **S3** 8 smoother steps are sufficient to reduce the error to a reasonable level before restriction.

5.3.4 Coarsest grid solvers

By using a NMG framework we are able to restrict our original problem on a large grid to a very coarse grid (e.g. 8×8). On this coarsest grid our aim is to solve the problem as accurately as possible, owing to the low computational cost, and so we need a designated solver for use solely on this coarsest grid. Here we present two coarsest grid solvers, based upon smoothers **S2** and **S3** from §5.2.4.

First proposed coarsest level solver C1. From §5.2.4, we know on the coarsest grid we are looking to solve the system of equations shown in (5.24) with coarse grid interval width $\boldsymbol{H}$ instead of the fine grid interval width $\boldsymbol{h}$. Equivalently we can express the system (5.24) in the following matrix form

$$\bar{\boldsymbol{A}}^H \boldsymbol{w}^H = \bar{\boldsymbol{F}}^H \tag{5.48}$$

where

$$\bar{\boldsymbol{A}}^H \in \mathbb{R}^{4(n-2)^2 \times 4(n-2)^2},\ \boldsymbol{w}^H,\ \bar{\boldsymbol{F}}^H \in \mathbb{R}^{4(n-2)^2 \times 1}$$

are given by

$$\bar{\boldsymbol{A}}^H = \begin{bmatrix} \boldsymbol{A}_1^H & \tilde{\boldsymbol{A}}_1^H & \boldsymbol{I}_2 & \boldsymbol{0} \\ \tilde{\boldsymbol{A}}_2^H & \boldsymbol{A}_2^H & \boldsymbol{0} & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{0} & \boldsymbol{B}_1^H & \tilde{\boldsymbol{B}}_1^H \\ \boldsymbol{0} & \boldsymbol{I}_2 & \tilde{\boldsymbol{B}}_2^H & \boldsymbol{B}_2^H \end{bmatrix}, \boldsymbol{w} = \begin{bmatrix} \boldsymbol{u}_1^H \\ \boldsymbol{u}_2^H \\ \boldsymbol{v}_1^H \\ \boldsymbol{v}_2^H \end{bmatrix}, \bar{\boldsymbol{F}} = \begin{bmatrix} \bar{\boldsymbol{F}}_1^H \\ \bar{\boldsymbol{F}}_2^H \\ \bar{\boldsymbol{G}}_1^H \\ \bar{\boldsymbol{G}}_2^H \end{bmatrix} \tag{5.49}$$

where

$$\boldsymbol{A}_m^H,\, \boldsymbol{B}_m^H \in \mathbb{R}^{(n-2)^2 \times (n-2)^2}$$

are the block tri-diagonal system matrices reflecting the coefficients of the

$$\left(u_m^H\right)_*^{(l+1)},\, \left(v_m^H\right)_*^{(l+1)}$$

terms at the various neighbouring pixels for each discrete interior point k respectively,

$$\tilde{\boldsymbol{A}}_m^H,\, \tilde{\boldsymbol{B}}_m^H \in \mathbb{R}^{(n-2)^2 \times (n-2)^2}$$

are the diagonal matrices corresponding to the

$$\left(u_t^H\right)_*^{(l+1)},\, \left(v_t^H\right)_*^{(l+1)}$$

terms in the

$$\left(u_m^H\right)_k^{(l+1)},\, \left(v_m^H\right)_k^{(l+1)}$$

equations respectively,

$$\boldsymbol{I}_2 = \beta \boldsymbol{I}$$

where $\boldsymbol{I}$ denotes the $(n-2)^2 \times (n-2)^2$ identity matrix and

$$\boldsymbol{u}_m^H,\, \boldsymbol{v}_m^H,\, \bar{\boldsymbol{F}}_m^H,\, \bar{\boldsymbol{G}}_m^H \in \mathbb{R}^{(n-2)^2 \times 1}$$

are the column vectors consisting of the displacements

$$\left(u_m^H\right)_k^{(l+1)},\, \left(v_m^H\right)_k^{(l+1)}$$

and RHS terms

$$\left(\bar{F}_m^H\right)_k^{(l+1)},\, \left(\bar{G}_m^H\right)_k^{(l+1)}$$

given by

$$\left(\bar{F}_m^H\right)_k = \sum_{s=1}^{2} \left(\partial_{u_m}^H T_{\boldsymbol{u}}^H\right)_k \left(\partial_{u_s}^H T_{\boldsymbol{u}}^H\right)_k \left(u_s^H\right)_k - \left(\partial_{u_m}^H T_{\boldsymbol{u}}^H\right)_k \Big[\left(T_{\boldsymbol{u}}^H\right)_k - \left(R^H\right)_k\Big]$$
$$\left(\bar{G}_m^H\right)_k = \sum_{s=1}^{2} \left(\partial_{v_m}^H R_{\boldsymbol{v}}^H\right)_k \left(\partial_{v_s}^H R_{\boldsymbol{v}}^H\right)_k \left(v_s^H\right)_k - \left(\partial_{v_m}^H R_{\boldsymbol{v}}^H\right)_k \Big[\left(R_{\boldsymbol{v}}^H\right)_k - \left(T^H\right)_k\Big]$$

where

$$k = (j-2)(n-1) + (i-1)$$

for $m = 1, 2$ and $i,\, j = 2, \ldots, n-1$. We then solve the matrix equation (5.48) using a direct method. In other words we solve

$$\boldsymbol{w}^H = \left[\bar{\boldsymbol{A}}^H\right]^{-1} \bar{\boldsymbol{F}}^H. \tag{5.50}$$

Second proposed coarsest level solver C2. Similar to what we did for **C1**, we can express the system (5.26) on the coarsest grid in the following matrix form

$$\hat{\boldsymbol{A}}^H \boldsymbol{w}^H = \bar{\boldsymbol{F}}^H \tag{5.51}$$

where $\boldsymbol{w}^H$, $\bar{\boldsymbol{F}}^H$ are as in (5.49) and

$$\hat{\boldsymbol{A}}^H \in \mathbb{R}^{4(n-2)^2 \times 4(n-2)^2}$$

has the following structure

$$\hat{\boldsymbol{A}}^H = \begin{bmatrix} \boldsymbol{A}_1^H & \boldsymbol{0} & \boldsymbol{I}_2 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}_2^H & \boldsymbol{0} & \boldsymbol{I}_2 \\ \boldsymbol{I}_2 & \boldsymbol{0} & \boldsymbol{B}_1^H & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_2 & \boldsymbol{0} & \boldsymbol{B}_2^H \end{bmatrix}$$

where

$$\boldsymbol{A}_m^H,\, \boldsymbol{B}_m^H \in \mathbb{R}^{(n-2)^2 \times (n-2)^2},\, \boldsymbol{u}_m^H,\, \boldsymbol{v}_m^H,\, \bar{\boldsymbol{F}}_m^H,\, \bar{\boldsymbol{G}}_m^H \in \mathbb{R}^{(n-2)^2 \times 1}$$

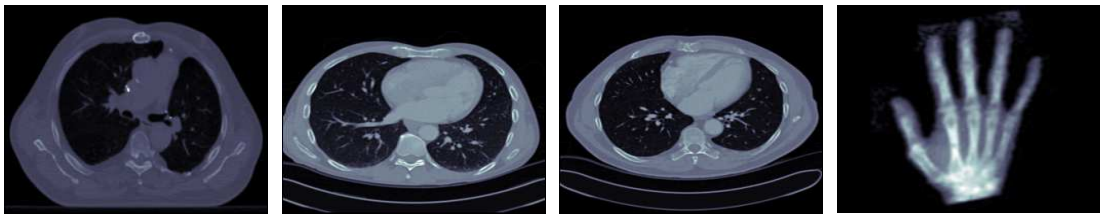
have the same structure as shown in **C1**, with RHS terms

$$\left(\bar{F}_m^H\right)_k^{(l+1)},\, \left(\bar{G}_m^H\right)_k^{(l+1)}$$

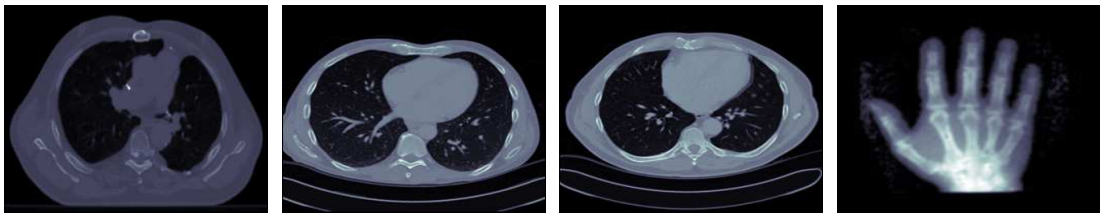
given by

$$\left(\bar{F}_m^H\right)_k = \Big[\left(\partial_{u_m}^H T_{\boldsymbol{u}}^H\right)^2\Big]_k \left(u_m^H\right)_k - \left(\partial_{u_m}^H T_{\boldsymbol{u}}^H\right)_k \Big[\left(T_{\boldsymbol{u}}^H\right)_k - \left(R^H\right)_k\Big],$$
$$\left(\bar{G}_m^H\right)_k = \Big[\left(\partial_{v_m}^H R_{\boldsymbol{v}}^H\right)^2\Big]_k \left(v_m^H\right)_k - \left(\partial_{v_m}^H R_{\boldsymbol{v}}^H\right)_k \Big[\left(R_{\boldsymbol{v}}^H\right)_k - \left(T^H\right)_k\Big].$$

Again we solve the matrix equation (5.51) using a direct method similar to (5.50).



(a) Reference R of Example 1. (b) Reference R of Example 2. (c) Reference R of Example 3. (d) Reference R of Example 4.



(e) Template T of Example 1. (f) Template T of Example 2. (g) Template T of Example 3. (h) Template T of Example 4.

Figure 5.3: Four pairs of test images.

Remark 5.3.2. *Our presented algorithms start from a fine grid for the registration process with the initial guess* $\boldsymbol{u} = \boldsymbol{0}$. *For small deformation (i.e.* $|\boldsymbol{u}|$ *is relatively small), such an initial guess is sufficient. However, for large deformation, a much better initialisation would be required. One way is to solve the registration model on the coarsest grid and work towards the finest grid in the so-called full multigrid framework. In many papers using a discretise-optimise approach, the similar use of a coarsest level to start the solution process to provide the initial guess on the finest level is called the multi-resolution scheme.*

5.4 Numerical results

Now we present some experimental results comparing three models, these are:

(i) A NMG scheme, similar to our proposed scheme, applied to a standard uni-directional diffusion model which we denote by **DNMG**;

(ii) Our proposed NMG (Algorithm 13) applied to our inverse consistent model equipped with smoother **S2** and solver **C1**, which we denote by **ICNMG1**;

(iii) Our proposed NMG (Algorithm 13) applied to our inverse consistent model equipped with smoother **S3** and solver **C2**, which we denote by **ICNMG2**.

Using these results we demonstrate how our new **ICNMG** models produce comparable results, both visually and numerically, to the **DNMG** model while maintaining non-folding results even in the case of a 'bad' parameter choice. In addition we also show how our simplified smoother **S3** in **ICNMG2** improves upon the CPU time, while

maintaining the same level of accuracy, compared with **ICNMG1** which uses the fully coupled smoother **S2**.

In order to gain a qualitative measure in the accuracy between the two models, we choose to use the structural similarity (SSIM) [108] and relative errors

$$Err_F = \frac{|T_{\boldsymbol{u}} - R|_2^2}{|R|_2^2}, \; Err_B = \frac{|R_{\boldsymbol{v}} - T|_2^2}{|T|_2^2}$$

corresponding to the forward and backward transformations respectively. Additionally in [18] it was shown the quantity

$$Q_{min} = \det\left(\nabla \boldsymbol{\varphi}\right) \tag{5.52}$$

can be used to indicate the presence of folding if (5.52) ≤ 0, likewise if (5.52) > 0 this indicates no folding is present and the transformation is therefore diffeomorphic. Moreover, we consider the NMG method to have converged only if one of the following criteria have been met:

(i) Average relative residual reaches $\varepsilon_1 = 10^{-2}$;

(ii) Maximum relative residual reaches $\varepsilon_2 = 10^{-2}$;

(iii) Number of NMG cycles reaches $\varepsilon_3 = 15$.

It should also be noted for our proposed **ICNMG** models, we only consider the NMG to have converged it both the forward and backward problems have converged according to the above stopping criteria. For all models we select the weighting parameter $\alpha = \frac{1}{15}$, and in our **ICNMG** models we set the second parameter to be $\beta = 10^4$. We performed our experiments on 3 sets of real lung CT images in addition to a synthetic hand X-ray image as shown in Figure 5.3. We also note in Tables 5.2-5.8 green Q_{min} values indicate no folding in the transformation, while red values indicate folding is present in the transformation.

Example 1 results. From Figure 5.4 we see the **DNMG** model, in addition to our **ICNMG** models, produce visually very similar results. This trend is backed up further by the results shown in Table 5.2, where we see near identical SSIM and relative error values. In addition we see our **ICNMG** models produce larger CPU times when compared with the **DNMG** model, however this increase is to be expected since our **ICNMG** models must solve additional equations. Moreover we also see our simplified smoother **S3**, which is used in our **ICNMG2** model, produces noticeably smaller CPU times when compared with our **ICNMG1** model (which uses the fully couple smoother **S2**) while maintaining the same level of accuracy. Also since our **ICNMG** models require both forward and backward problems to converge, we see a slight increase in the number of NMG cycles required when compared with the **DNMG** model. This pattern of results is also seen in Table 5.3 where again all 3 models produce similar results with

our **ICNMG** models requiring an additional NMG cycle to converge plus larger CPU times, with our **ICNMG2** model being significantly faster than our **ICNMG1** model. In all cases we see each of the 3 models produce positive Q_{min} values which indicates no folding is present in the transformations.

Example 2 results. In Example 2, wee see the same pattern of results which we saw in Example 1. Namely near identical results both visually (Figure 5.5) and numerically (Tables 5.4 and 5.5) with larger CPU times for our **ICNMG** models, and our **ICNMG2** model much faster than our **ICNMG1** model. In addition all 3 models produce non-folding results in all cases. However when considering the 'bad' parameter case $\alpha = \frac{1}{25}$ in Table 5.6, we see the **DNMG** model produces negative Q_{min} values in 3 out of the 4 cases whereas both of our **ICNMG** models maintain the physical integrity of the transformation while achieving the same level of accuracy in all 4 cases. An example of how the mesh plots of the transformations from the **DNMG** model and our **ICNMG2** model for the 128^2 example from Table 5.6 can be seen in Figure 5.2. Here we see the mesh from our **ICNMG2** model is much smoother than that from the **DNMG** model.

Example 3 results. From Figure 5.6 and Tables 5.7 and 5.8 we see the same trend in results which was present in Examples 1 and 2, while we again see all cases produce non-folding transformations. **Example 4 Results**. Looking at Figure 5.7 we see that visually all models produce very similar results. However if we look at Tables 5.9 and 5.10 we see our **ICNMG** models produce slightly larger error values when compared with the **DNMG**, although these differences do no show visually. With regard to CPU time we see exactly the same patterns which we saw in Examples 1-3.

Testing of parameter sensitivity for ICNMG2 model. Here we perform a test on how robust our **ICNMG2** model is to the choice of parameters α and β. To do this we tracked the $SSIM$ and Q_{min} values across a total of 25 different sets of parameter values, in other words we tested all combinations resulting from the parameters

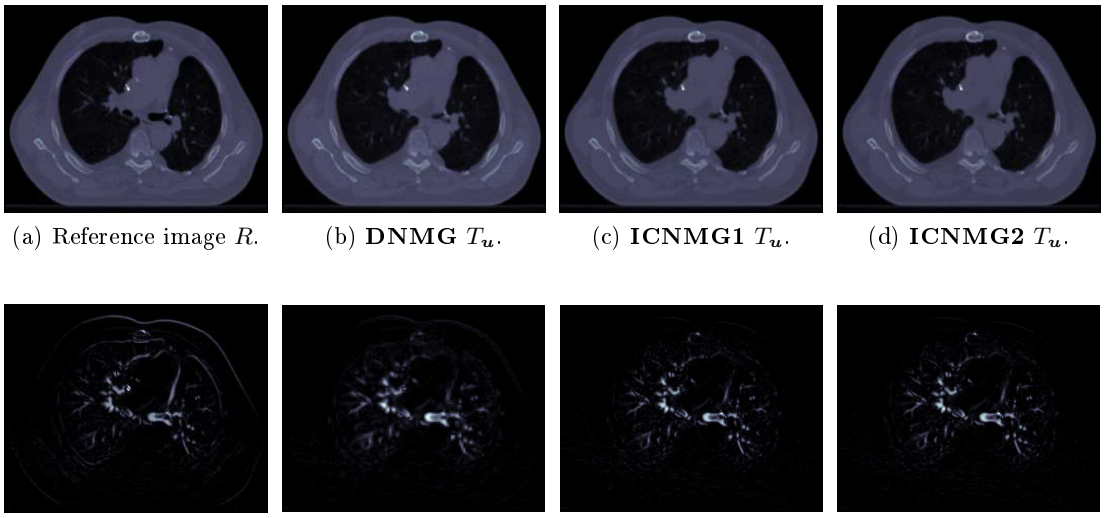
$$\alpha = \frac{1}{10}, \frac{1}{15}, \frac{1}{20}, \frac{1}{25}, \frac{1}{30}, \beta = 0, 10^3, 10^4, 10^5, 10^6.$$

the results of which can be seen in Figures 5.8 and 5.9 respectively. In addition we remark we have included a simulation for the **DNMG** model in our tests by considering the parameter $\beta = 0$. From Figure 5.8 we see our **ICNMG2** model maintains very similar $SSIM$ values when compared with the **DNMG** model ($\beta = 0$ column), and there is little variation in the values as the parameter β is varied in our **ICNMG2** model. However the advantage of our **ICNMG2** model is shown more clearly in Figure 5.9 where we have tracked the Q_{min} values across the different parameter tests, here red indicates $Q_{min} < 0$ while green indicates $Q_{min} > 0$. From this figure we see our **ICNMG2** is robust to folding for a much larger range of α values when compared with the diffusion model which has a much more limited range of viable α choices.

Test on NMG Efficiency. In addition to the physicality of the transformation obtained from the image registration process, we are also concerned with the fast solution via a NMG framework. For this reason it is important to achieve optimal efficiency for the NMG scheme, which should be of order $\mathcal{O}(N\log(N))$ where $N = n^2$. This means for an optimal NMG scheme, we would expect to see an increase in CPU time by a factor of approximately 4.5 when increasing both image dimensions by a factor of 2. From Table 5.11 we see all three discussed NMG schemes (i.e. **DNMG**, **ICNMG1** and **ICNMG2**) all exhibit near optimal efficiency.

5.5 Summary

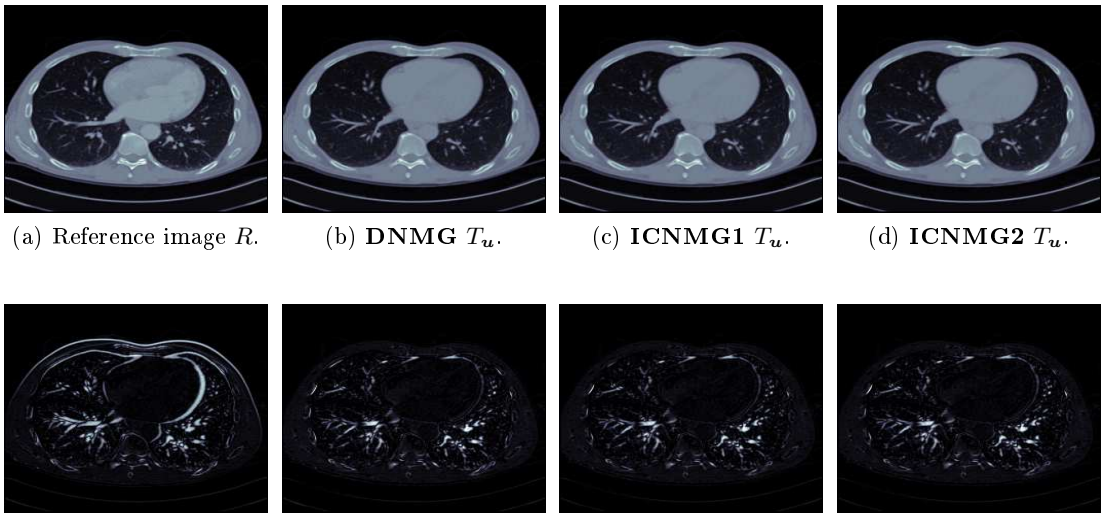
In this chapter we first explained how many standard variational registration models do no place any emphasis on maintaining the physical accuracy of the transformations, thus potentially leading to physically inaccurate transformations with folding. Next we explained how inverse consistent models, such as the Christensen-Johnson model proposed in [28], can help improve robustness to folding. We also mentioned how the model in [28] is impractical for real medical image problems owing to the extensive computational cost resulting from solving the associated minimisation problem. In order to help avoid this problem, we first proposed a linearisation of the inverse consistency constraint from the Christensen-Johnson model to remove the additional non-linearities arising from this term when compared with typical diffusion type models, in addition to alleviating the computational cost of directly computing the inverse displacements. Next we proposed the use of a fast NMG framework, based upon the scheme proposed by Chumchob and Chen in [33], along with 3 potential smoother schemes to further reduce the computational workload of the proposed inverse consistent model. In addition we also performed an analysis of the 3 proposed smoothers to determine their suitability for use in the NMG scheme, and how they can impact the convergence of the NMG. Next we showed, using 3 sets of real lung CT images and 1 set of synthetic hand X-ray images, how our proposed inverse consistent model maintains the same level of accuracy as a uni-directional diffusion model using a similar NMG scheme, while being robust to parameter choice and folding even in the case of a 'bad' weighting parameter value which causes folding in the transformation obtained from the diffusion model.



(a) Reference image R. (b) **DNMG** $T_{\boldsymbol{u}}$. (c) **ICNMG1** $T_{\boldsymbol{u}}$. (d) **ICNMG2** $T_{\boldsymbol{u}}$.

(e) $|T_{\boldsymbol{u}} - R|$, $Err = 0.38\,\%$. (f) **DNMG** $|T_{\boldsymbol{u}} - R|$, $Err = 0.27\,\%$. (g) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 0.28\,\%$. (h) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 0.28\,\%$.

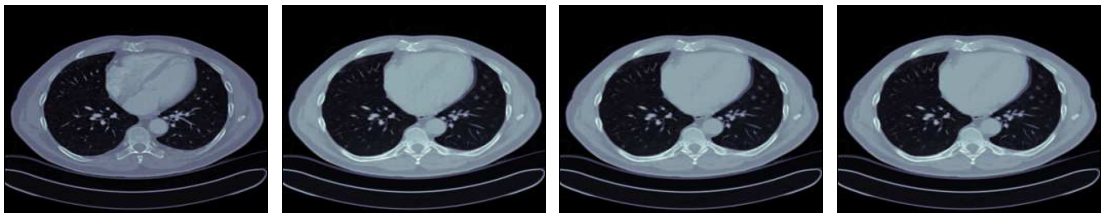
Figure 5.4: Example 1: Registration of Figure 5.3 (a) and Figures 5.3 (e) of size 256×256 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **DNMG**, **ICNMG1** and **ICNMG2** models respectively, while images (f), (g) and (h) show the respective final errors.



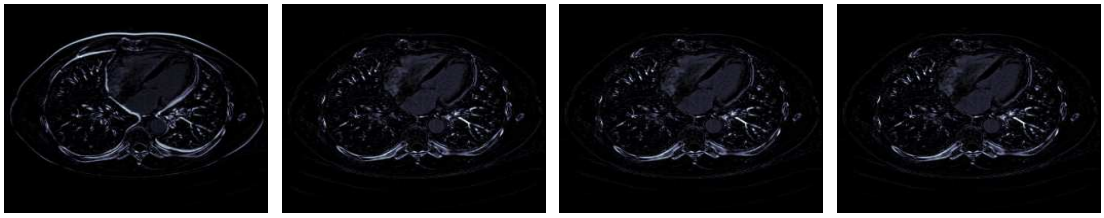
(a) Reference image R. (b) **DNMG** $T_{\boldsymbol{u}}$. (c) **ICNMG1** $T_{\boldsymbol{u}}$. (d) **ICNMG2** $T_{\boldsymbol{u}}$.

(e) $|T_{\boldsymbol{u}} - R|$, $Err = 1.07\,\%$. (f) **DNMG** $|T_{\boldsymbol{u}} - R|$, $Err = 0.40\,\%$. (g) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 0.42\,\%$. (h) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 0.42\,\%$.

Figure 5.5: Example 2: Registration of Figure 5.3 (b) and Figures 5.3 (f) of size 256×256 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **DNMG**, **ICNMG1** and **ICNMG2** models respectively, while images (f), (g) and (h) show the respective final errors.

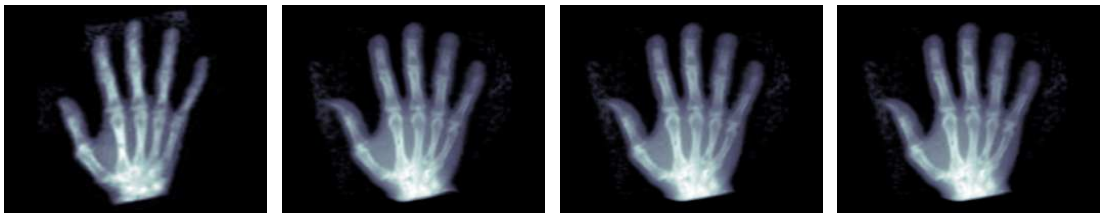


(a) Reference image R. (b) **DNMG** $T_{\boldsymbol{u}}$. (c) **ICNMG1** $T_{\boldsymbol{u}}$. (d) **ICNMG2** $T_{\boldsymbol{u}}$.

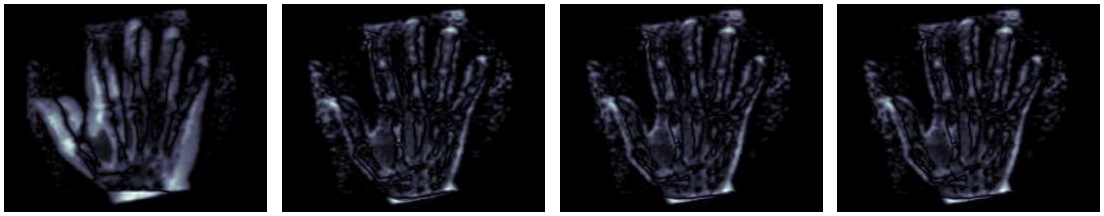


(e) $|T_{\boldsymbol{u}} - R|$, $Err = 1.05\,\%$. (f) **DNMG** $|T_{\boldsymbol{u}} - R|$, $Err = 0.31\,\%$. (g) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 0.32\,\%$. (h) **ICNMG2** $|T_{\boldsymbol{u}} - R|$, $Err = 0.32\,\%$.

Figure 5.6: Example 3: Registration of Figure 5.3 (c) and Figures 5.3 (g) of size 256×256 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **DNMG**, **ICNMG1** and **ICNMG2** models respectively, while images (f), (g) and (h) show the respective final errors.

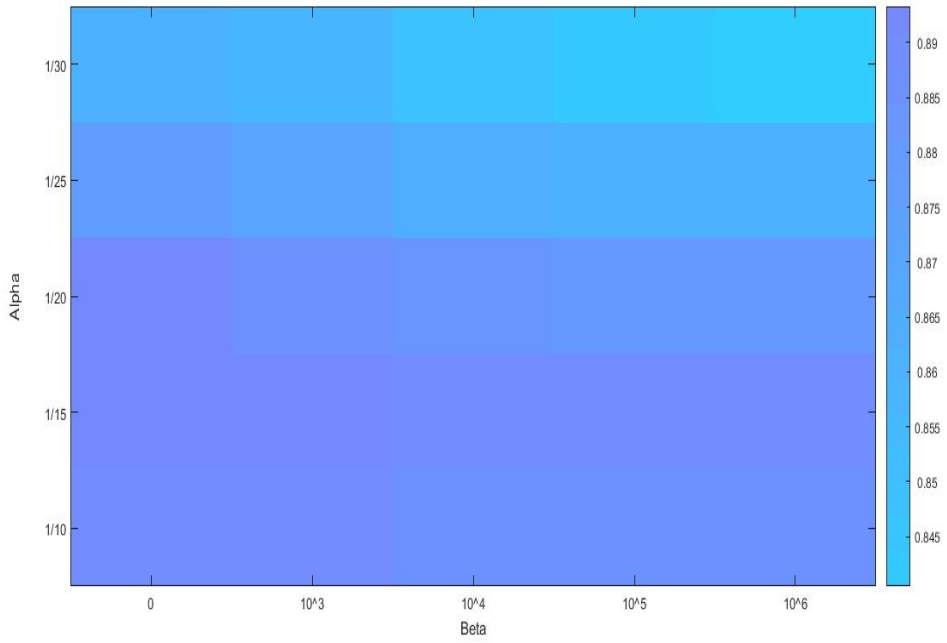


(a) Reference image R. (b) **DNMG** $T_{\boldsymbol{u}}$. (c) **ICNMG1** $T_{\boldsymbol{u}}$. (d) **ICNMG2** $T_{\boldsymbol{u}}$.

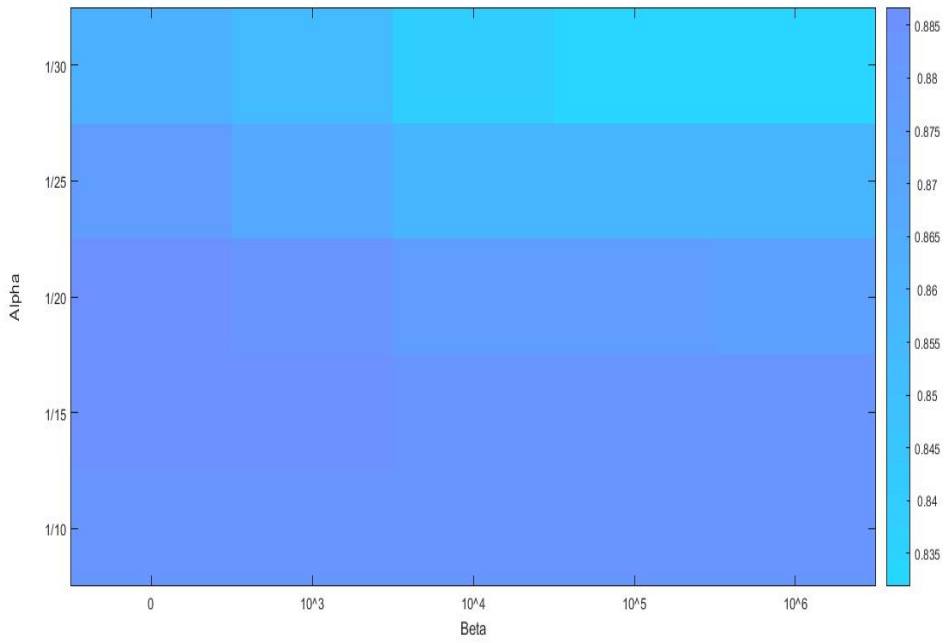


(e) $|T_{\boldsymbol{u}} - R|$, $Err = 13.24\,\%$. (f) **DNMG** $|T_{\boldsymbol{u}} - R|$, $Err = 1.30\,\%$. (g) **ICNMG1** $|T_{\boldsymbol{u}} - R|$, $Err = 1.62\,\%$. (h) **ICNMG2** $|T_{\boldsymbol{u}} - R|$, $Err = 1.62\,\%$.

Figure 5.7: Example 4: Registration of Figure 5.3 (d) and Figures 5.3 (h) of size 256×256 by three methods with initial error shown by image (e). Images (b), (c) and (d) show the deformed template images obtained using the **DNMG**, **ICNMG1** and **ICNMG2** models respectively, while images (f), (g) and (h) show the respective final errors.

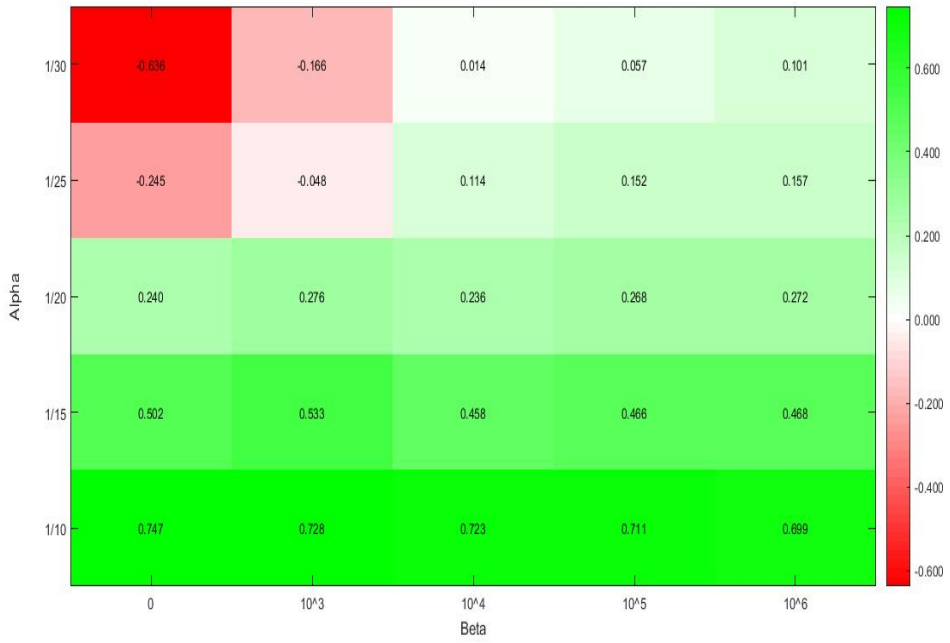


(a) Heat map of $SSIM$ values over a range of parameter choices α, β for the forward problem.

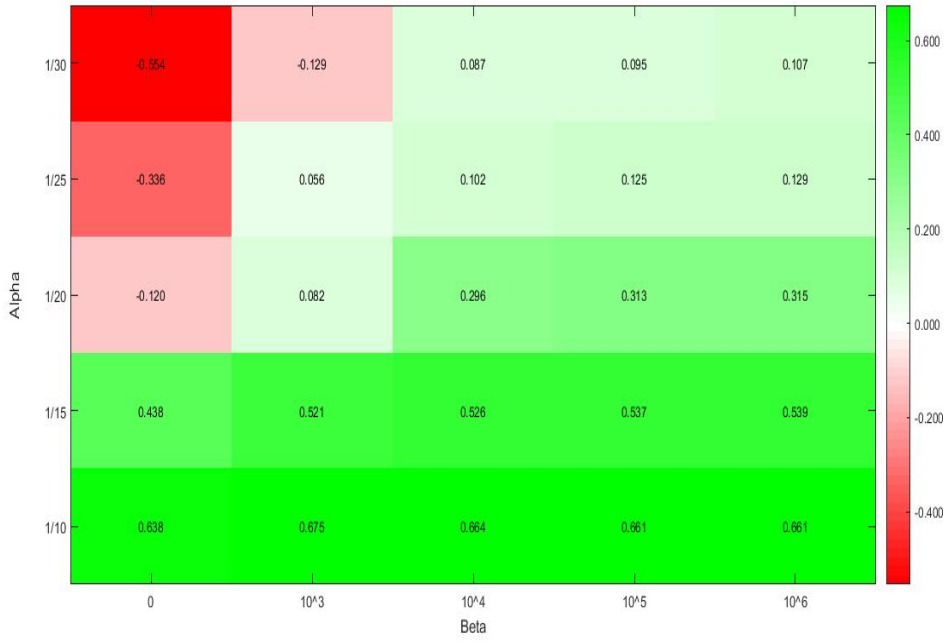


(b) Heat map of $SSIM$ values over a range of parameter choices α, β for the backward problem.

Figure 5.8: Comparison of how the $SSIM$ values vary with different choices of the parameters α and β for Example 2.



(a) Heat map of Q_{min} values over a range of parameter choices α, β for the forward problem.



(b) Heat map of Q_{min} values over a range of parameter choices α, β for the backward problem.

Figure 5.9: Comparison of how the Q_{min} values vary with different choices of the parameters α and β for Example 2.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**
128^2	0.915	0.35	0.938	0.22	1	0.167	0.553	0.938	0.22	2	1.498	0.489	0.938	0.22	2	0.879	0.489
256^2	0.914	0.38	0.935	0.27	1	0.822	0.673	0.932	0.28	2	5.155	0.654	0.933	0.28	2	3.031	0.654
512^2	0.939	0.37	0.953	0.27	1	4.082	0.669	0.949	0.28	2	24.557	0.658	0.949	0.28	2	14.180	0.658
1024^2	0.958	0.37	0.967	0.27	1	18.818	0.667	0.964	0.29	2	111.034	0.656	0.964	0.29	2	66.814	0.656

Table 5.2: Example 1: Comparison of forward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**
128^2	0.915	0.34	0.940	0.17	1	0.204	0.654	0.939	0.22	2	1.498	0.786	0.939	0.22	2	0.879	0.786
256^2	0.914	0.37	0.936	0.22	1	0.874	0.573	0.934	0.27	2	5.155	0.718	0.934	0.27	2	3.031	0.719
512^2	0.939	0.36	0.953	0.22	1	4.046	0.639	0.949	0.27	2	24.557	0.695	0.949	0.27	2	14.180	0.695
1024^2	0.958	0.36	0.968	0.22	1	17.935	0.633	0.965	0.28	2	111.034	0.686	0.965	0.28	2	66.814	0.686

Table 5.3: Example 1: Comparison of backward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	***Q*$_{min}$**
128^2	0.808	1.02	0.892	0.37	2	0.415	0.451	0.891	0.37	2	1.582	0.353	0.890	0.37	2	0.640	0.241
256^2	0.767	1.07	0.871	0.40	2	1.512	0.250	0.868	0.42	2	5.202	0.157	0.868	0.42	2	3.025	0.024
512^2	0.779	1.08	0.868	0.41	2	6.819	0.519	0.866	0.43	2	24.572	0.423	0.866	0.43	2	14.232	0.423
1024^2	0.828	1.08	0.892	0.40	2	31.895	0.520	0.891	0.43	2	111.561	0.413	0.891	0.43	2	66.537	0.413

Table 5.4: Example 2: Comparison of forward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err$_F$*** (%)	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}
128^2	0.808	1.00	0.886	0.36	2	0.479	0.361	0.886	0.36	2	1.582	0.155	0.885	0.36	2	0.640	0.073
256^2	0.767	1.05	0.861	0.38	2	1.561	0.212	0.861	0.41	2	5.202	0.220	0.860	0.41	2	3.025	0.167
512^2	0.779	1.06	0.862	0.40	2	7.054	0.419	0.861	0.42	2	24.572	0.366	0.861	0.42	2	14.232	0.366
1024^2	0.828	1.06	0.889	0.40	2	31.370	0.405	0.890	0.42	2	111.561	0.350	0.890	0.42	2	66.537	0.350

Table 5.5: Example 2: Comparison of backward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err$_F$*** (%)	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}
128^2	0.808	1.02	0.872	0.36	2	0.426	−0.245	0.896	0.36	2	1.521	0.360	0.886	0.36	2	0.821	0.114
256^2	0.767	1.07	0.855	0.32	4	2.182	−0.374	0.874	0.36	2	5.255	0.220	0.871	0.36	2	3.355	0.316
512^2	0.779	1.08	0.876	0.34	2	6.907	−0.141	0.872	0.36	2	24.525	0.098	0.871	0.36	2	15.225	0.214
1024^2	0.828	1.08	0.900	0.32	2	33.889	0.214	0.896	0.36	2	111.118	0.168	0.895	0.36	2	73.118	0.240

Table 5.6: Example 2: Comparison of forward registrations between three methods on different image sizes for a 'bad' parameter value $\alpha = \frac{1}{25}$.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err$_F$*** (%)	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}	***SSIM***	***Err$_F$*** (%)	***NMG***	***CPU*** (s)	Q_{min}
128^2	0.847	0.94	0.908	0.34	2	0.324	0.230	0.910	0.37	2	1.414	0.259	0.900	0.39	2	0.646	0.169
256^2	0.805	1.05	0.899	0.31	2	1.418	0.513	0.897	0.32	2	5.147	0.467	0.896	0.32	2	3.007	0.416
512^2	0.805	1.08	0.884	0.32	2	6.941	0.481	0.882	0.32	2	24.795	0.491	0.882	0.32	2	14.195	0.490
1024^2	0.842	1.08	0.901	0.32	2	33.210	0.411	0.902	0.32	2	111.887	0.589	0.902	0.32	2	66.789	0.588

Table 5.7: Example 3: Comparison of forward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**
128^2	0.847	1.01	0.915	0.35	2	0.391	0.350	0.912	0.40	2	1.414	0.168	0.904	0.42	2	0.646	0.012
256^2	0.805	1.12	0.899	0.34	2	1.485	0.525	0.899	0.34	2	5.147	0.489	0.898	0.34	2	3.007	0.461
512^2	0.805	1.16	0.882	0.34	2	6.930	0.467	0.882	0.35	2	24.795	0.416	0.882	0.35	2	14.195	0.416
1024^2	0.842	1.16	0.899	0.34	2	33.301	0.440	0.902	0.35	2	111.887	0.435	0.902	0.35	2	66.789	0.435

Table 5.8: Example 3: Comparison of backward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**
128^2	0.552	13.14	0.782	1.35	1	0.157	0.517	0.773	1.62	1	0.939	0.448	0.773	1.62	1	0.570	0.449
256^2	0.584	13.24	0.779	1.30	1	0.756	0.483	0.767	1.62	1	2.738	0.421	0.767	1.61	1	1.456	0.421
512^2	0.634	13.25	0.795	1.29	1	4.039	0.477	0.789	1.62	1	12.742	0.418	0.787	1.62	1	7.616	0.418
1024^2	0.693	13.25	0.821	1.29	1	18.042	0.475	0.814	1.62	1	57.839	0.419	0.814	1.62	1	36.321	0.419

Table 5.9: Example 4: Comparison of forward registrations between three methods on different image sizes.

	Initial		DNMG					ICNMG1					ICNMG2				
Image Size (n^2)	***SSIM***	***Err*$_F$ (%)**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**	***SSIM***	***Err*$_F$ (%)**	***NMG***	***CPU* (s)**	**Q_{min}**
128^2	0.552	9.75	0.776	1.00	1	0.167	0.402	0.765	1.21	1	0.939	0.198	0.765	1.20	1	0.570	0.198
256^2	0.584	9.83	0.768	0.96	1	0.761	0.371	0.756	1.20	1	2.738	0.143	0.756	1.20	1	1.456	0.143
512^2	0.639	9.84	0.793	0.96	1	4.051	0.369	0.783	1.20	1	12.742	0.156	0.783	1.20	1	7.616	0.156
1024^2	0.693	9.84	0.823	0.96	1	18.52	0.364	0.817	1.20	1	57.839	0.156	0.817	1.20	1	36.321	0.156

Table 5.10: Example 4: Comparison of backward registrations between three methods on different image sizes.

			DNMG		ICNMG1		ICNMG2	
Image Size (n^2)	**Image Example**	$\boldsymbol{\alpha}$	***CPU*** **(*s*)**	**Ratio**	***CPU*** **(*s*)**	**Ratio**	***CPU*** **(*s*)**	**Ratio**
128^2	Example 2 (Forward)	$\frac{1}{15}$	0.415	–	1.582	–	0.640	–
256^2			1.512	3.643	5.202	3.288	3.025	4.727
512^2			6.819	4.510	24.572	4.724	14.232	4.705
1024^2			31.895	4.677	111.561	4.540	66.537	4.675

Table 5.11: Test on optimal complexity in CPU time ratio for three NMG methods. The optimal ratio is approximately 4.5 for an $\mathcal{O}(N \log N)$ NMG method (with $N = n^2$).

Chapter 6

Preliminary validation of two non-folding 3D registration models for use in oncology

In the previous chapters we have, so far, only formulated the registration problem in 2D. However, since most medical applications require the registration of 3D (or even 4D) images, we must extend the previously discussed models to the 3D case. In this chapter we begin with a review of how the registration problem is constructed in the 3D case, before showing how the models discussed in the previous chapters can be extended to 3D. After this we describe how these 3D models are solved in addition to introducing, and analysing, a fast NMG framework to reduce the CPU cost of solving these models in 3D. Next we present some preliminary results measuring how accurately our proposed models deform and match contoured features of lung CT images when compared with the commercial Eclipse software.

6.1 Introduction

Suppose we are given a 3D reference and template image, which we denote by R, T respectively, defined on a subspace of $\mathbb{R}^3$ which is denoted by Ω, i.e.

$$R,\, T \in \Omega \subset \mathbb{R}^3.$$

Moreover, let us assume the image domain Ω is defined by the unit cube, i.e.

$$\Omega = [0,1]^3.$$

Then we formulate the 3D variational registration problem as the minimisation of the following joint energy functional

$$\min_{\boldsymbol{u}} \left\{ E(\boldsymbol{u}) = \mathscr{D}\big(R, T, \boldsymbol{u}\big) + \alpha \mathscr{R}\big(\boldsymbol{u}\big) \right\} \tag{6.1}$$

where $\mathscr{D}\big(R, T, \boldsymbol{u}\big)$ denotes the similarity measure between R and T, $\mathscr{R}\big(\boldsymbol{u}\big)$ denotes the regularisation of $\boldsymbol{u}$ required to overcome the ill-posedness of the minimisation problem (6.1), $\alpha \in \mathbb{R}^+$ is a weighting parameter between $\mathscr{D}$ and $\mathscr{R}$ and

$$\boldsymbol{u} \equiv \boldsymbol{u}(\boldsymbol{x}) = \big[u_1(\boldsymbol{x}), u_2(\boldsymbol{x}), u_3(\boldsymbol{x})\big]^T \in \Omega,\ \boldsymbol{x} = \big[x_1, x_2, x_3\big]^T \in \Omega$$

denotes the 3D displacement field. Furthermore, let us also assume the images R and T are mono-modal, then we select the SSD measure to be the distance term given by

$$\mathscr{D}^{SSD}\big(R, T, \boldsymbol{u}\big) = \frac{1}{2} \int_{\Omega} |T_{\boldsymbol{u}} - R|^2 \ d\Omega \tag{6.2}$$

where

$$T_{\boldsymbol{u}} \equiv T\big(\boldsymbol{x} + \boldsymbol{u}\big),\ R \equiv R(\boldsymbol{x})$$

and $|\cdot|$ denotes the Euclidean norm.

In oncology image registration plays a vital role in the effective planning and treatment of lung cancer, thus improvements in the accuracy of registration models is a necessity. Typically there are four main applications of image registration in radiotherapy applications, these are:

(i) Multi-modal image registration;

(ii) 4D dose accumulation [1, 42, 58, 93, 114, 115, 134, 137, 154];

(iii) Lung ventilation imaging [83, 135];

(iv) Anatomic image segmentation [23, 53, 70, 110].

Multi-modal image registration. When treating lung cancer, it is common for multi-modal imaging to be used for treatment. For example we can overlay functional imaging data (e.g. CT or PET), which provides information regarding the activity of biological processes, onto high quality anatomical imaging data (e.g. X-ray CT) which reveals individual structures. For this reason multi-modal image registration models are required to ensure an accurate evaluation of the irradiated dose of the tumour and organs which are at risk.

4D dose accumulation. As a result of the patient breathing during a CT scan of the lungs, both rigid body and non-rigid deformation of organs takes place. This means it is not possible to compare dose maps from one phase to another because of voxel movement. However this problem can be overcome by using image registration to match

all phases to a single reference phase and applying the found deformations to the dose maps so they are all aligned to the reference phase. From this all of the deformed dose maps can be combined to produce a 4D dose accumulation map which indicates the total dose applied to each region of the lungs.

Lung ventilation imaging. Another useful piece of information which can be extracted from lung CT images after registration has been performed is how air content changes in the lungs due to ventilation. This information allows us to see areas of the lungs which are no longer functioning, or functioning poorly. To obtain these ventilation maps we use the deformation fields obtained from the registration step, and then either compute the Jacobian or Hounsfield unit change metric. Given the deformation field

$$\boldsymbol{u} \equiv \left[u_1(\boldsymbol{x}), u_2(\boldsymbol{x}), u_3(\boldsymbol{x})\right]^T \in \Omega \subset \mathbb{R}^3,\ \boldsymbol{x} = \left[x_1, x_2, x_3\right]^T \in \Omega$$

the Jacobian is given by

$$\boldsymbol{u}_J(\boldsymbol{x}) = \begin{vmatrix} 1+\partial_{x_1}u_1(\boldsymbol{x}) & \partial_{x_2}u_1(\boldsymbol{x}) & \partial_{x_3}u_1(\boldsymbol{x}) \\ \partial_{x_1}u_2(\boldsymbol{x}) & 1+\partial_{x_2}u_2(\boldsymbol{x}) & \partial_{x_3}u_2(\boldsymbol{x}) \\ \partial_{x_1}u_3(\boldsymbol{x}) & \partial_{x_2}u_3(\boldsymbol{x}) & 1+\partial_{x_3}u_3(\boldsymbol{x}) \end{vmatrix}. \tag{6.3}$$

Alternatively we can compute the Hounsfield unit change metric, using the following

$$\boldsymbol{u}_{HU}(\boldsymbol{x}) = \frac{HU_{ex}(\boldsymbol{x}) - HU_{in}(\boldsymbol{x}+\boldsymbol{u})}{HU_{in}(\boldsymbol{x}+\boldsymbol{u}) + 1000} \tag{6.4}$$

where $HU_{ex}(\boldsymbol{x})$ denotes the Hounsfield unit value at voxel $\boldsymbol{x}$ of the peak exhale image and $HU_{in}(\boldsymbol{x}+\boldsymbol{u})$ denotes the corresponding Hounsfield unit value of the peak inhale image. Both metrics (6.3) and (6.4) give information about regional volume change.

Anatomic image segmentation. During radiotherapy treatments contours are drawn, typically by hand, directly onto the CT scans to highlight important regions such as the gross tumour volume (GTV) and planning target volume (PTV). However throughout the treatment process, the patients' anatomy will change in response to the treatment, thus the contours will need to be re-drawn onto the new set of scans. Since the contouring process is a very difficult and time consuming process for the physician (possibly requiring several hours to complete a single CT set), it would therefore be beneficial to have an automatic method which can be used to update the original contours when needed. Indeed this task can be achieved by using image registration. Given the contours from the original CT scan, we use registration to deform the original images to the new images. Then, using the corresponding deformation fields, we can deform the original contours to obtain contours on the new images.

One of the biggest challenges when developing image registration models for use in the treatment of lung cancer is the lack of ground truth data available to validate the models. In this chapter we propose two 3D registration models whose main goal is to

produce physically accurate deformations with no folding, in addition to introducing a standard 3D diffusion model for comparison.

The remainder of this chapter will be set out as followed. In §6.2 we introduce the 3D extensions of the models discussed in the previous chapters 4 and 5, in addition to outlining how we solve these models numerically in §6.3. Next in §6.4 we introduce the fast NMG framework which will be implemented for each model, alongside performing an analysis on each of the key components of the NMG framework. Then in §6.5 we present some preliminary results, performed on eight examples taken from the Hugo dataset [80], comparing the proposed 3D models with the state of the art Eclipse software. Finally we present a chapter summary in §6.6

6.2 Review of 3D registration models and numerical implementation

We now briefly discuss how the models discussed in Chapters 4 and 5 are formulated in the 3D case. After this we describe how to solve these 3D models numerically.

6.2.1 3D diffusion model

Our first proposed model (as seen in §4.3), is based upon the diffusion model [13, 15, 16, 27, 30, 33, 43, 46, 47, 105]. In this model we use the SSD similarity measure (6.2) and combine it with the following 3D diffusion regulariser

$$\mathscr{R}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega}\sum_{s=1}^{3}|\nabla_3 u_s|^2\,d\Omega \tag{6.5}$$

where

$$\nabla_3 \equiv \left(\partial_{x_1}, \partial_{x_2}, \partial_{x_3}\right)^T$$

denotes the 3D spatial gradient operator. Combining the regulariser (6.5) with the similarity measure (6.2) in the minimisation problem (6.1) leads to the 3D diffusion model, which is given by the following

$$\min_{\boldsymbol{u}}\left\{E^{Diff}(\boldsymbol{u}) = \frac{1}{2}\int_{\Omega}|T_{\boldsymbol{u}} - R|^2 + \alpha\sum_{s=1}^{3}|\nabla_3 u_s|^2\,d\Omega\right\}. \tag{6.6}$$

To solve the minimisation problem (6.6), we use an optimise-discretise approach which involves deriving and solving the EL equations of the energy functional in (6.6). It can be shown the EL equations, corresponding to the 3D diffusion model (6.6), are given by

$$-\alpha\Delta_3 u_m + F_m^{Diff}(\boldsymbol{u}) = 0 \tag{6.7}$$

with Neumann boundary conditions

$$\nabla_3 u_m \cdot \boldsymbol{n} = 0$$

where $\boldsymbol{n}$ denotes the outward unit normal,

$$\Delta_3 \equiv \partial_{x_1 x_1} + \partial_{x_2 x_2} + \partial_{x_3 x_3}$$

denotes the 3D Laplace operator and

$$F_m^{Diff}\big(\boldsymbol{u}\big) = \partial_{u_m} T_{\boldsymbol{u}}\big[T_{\boldsymbol{u}} - R\big] \tag{6.8}$$

denote the force terms for $m = 1, 2, 3$.

6.2.2 3D constrained diffusion model

For any real life medical applications, a necessity for any deformation field obtained using image registration is it must be physical. In others words it is crucial the deformation obtained contains no folding. In §4.4 we introduced a constrained version of the diffusion model to enforce a positive value of the determinant of the gradient of the transformation

$$\boldsymbol{\varphi} \equiv \boldsymbol{\varphi}(\boldsymbol{x}) = \boldsymbol{x} + \boldsymbol{u}$$

which is computed using

$$\det\big(\nabla_3 \boldsymbol{\varphi}\big) > 0.$$

We can also apply a similar constraint to the EL equations of the 3D diffusion model shown in (6.7), this then gives the following

$$-\alpha \Delta_3 u_m + F_m^{Diff}\big(\boldsymbol{u}\big) = 0, \; s.t. \; \det\big(\nabla_3 \boldsymbol{\varphi}\big) > 0 \tag{6.9}$$

with Neumann boundary conditions, where $F_m^{Diff}(\boldsymbol{u})$ are the force terms from (6.8) and

$$\det\left(\nabla_3\boldsymbol{\varphi}\right) = \begin{vmatrix} \partial_{x_1}\varphi_1 & \partial_{x_2}\varphi_1 & \partial_{x_3}\varphi_1 \\ \partial_{x_1}\varphi_2 & \partial_{x_2}\varphi_2 & \partial_{x_3}\varphi_2 \\ \partial_{x_1}\varphi_3 & \partial_{x_2}\varphi_3 & \partial_{x_3}\varphi_3 \end{vmatrix}$$

$$\begin{aligned} &= \partial_{x_1}\varphi_1\partial_{x_2}\varphi_2\partial_{x_3}\varphi_3 - \partial_{x_1}\varphi_1\partial_{x_3}\varphi_2\partial_{x_2}\varphi_3 - \partial_{x_3}\varphi_1\partial_{x_1}\varphi_2\partial_{x_3}\varphi_3 \\ &+ \partial_{x_2}\varphi_1\partial_{x_3}\varphi_2\partial_{x_3}\varphi_1 + \partial_{x_3}\varphi_1\partial_{x_1}\varphi_2\partial_{x_2}\varphi_3 - \partial_{x_3}\varphi_1\partial_{x_2}\varphi_2\partial_{x_1}\varphi_3 \end{aligned}$$

$$\begin{aligned} &= \big(1+\partial_{x_1}u_1\big)\big(1+\partial_{x_2}u_2\big)\big(1+\partial_{x_3}u_3\big) - \big(1+\partial_{x_1}u_1\big)\partial_{x_3}u_2\partial_{x_2}u_3 \\ &- \partial_{x_2}u_1\partial_{x_1}u_2\big(1+\partial_{x_3}u_3\big) + \partial_{x_2}u_1\partial_{x_3}u_2\partial_{x_1}u_3 + \partial_{x_3}u_1\partial_{x_1}u_2\partial_{x_2}u_3 \\ &- \partial_{x_3}u_1\big(1+\partial_{x_2}u_2\big)\partial_{x_1}u_3. \end{aligned} \tag{6.10}$$

Subsequently we proposed an extension to our constrained diffusion model in §4.4.4 to improve accuracy and robustness to the weighting parameter α. Again, this treatment can also be applied to the 3D model (6.9), which we demonstrate in §6.4.

6.2.3 3D inverse consistent model

In Chapter 5 we proposed an alternate model with the aim of achieving diffeomorphic (i.e. non-folding) deformations, we did this through the use of an inverse consistent model [26, 28, 66, 113, 147]. For the 3D case, the proposed inverse consistent model (based upon the Christensen-Johnson inverse consistent model [28]) is given by

$$\begin{aligned} \min_{\boldsymbol{u},\,\boldsymbol{v}} \Bigg\{ E^{IC}(\boldsymbol{u},\boldsymbol{v}) = \frac{1}{2}\int_\Omega |T_{\boldsymbol{u}} - R|^2 + |R_{\boldsymbol{v}} - T|^2 + \alpha\left[|\nabla_3\boldsymbol{u}|^2 + |\nabla_3\boldsymbol{v}|^2\right] \\ + \beta\left[|\boldsymbol{u}+\boldsymbol{v}|^2 + |\boldsymbol{v}+\boldsymbol{u}|^2\right]\, d\Omega \Bigg\} \end{aligned} \tag{6.11}$$

where $\boldsymbol{u}$, $\boldsymbol{v}$ denote the forward and backward displacements,

$$R \equiv R(\boldsymbol{x}),\; T \equiv T(\boldsymbol{x})$$

denote the reference images for the forward ($T \to R$) and backward ($R \to T$) problems,

$$T_{\boldsymbol{u}} \equiv T\big(\boldsymbol{x}+\boldsymbol{u}\big),\; R_{\boldsymbol{v}} \equiv R\big(\boldsymbol{x}+\boldsymbol{v}\big)$$

denote the deformed template images for the forward and backward problems respectively and

$$0 \leq \beta \in \mathbb{R},\; \alpha \in \mathbb{R}^+$$

are weighting parameters. Equivalently we can split the joint minimisation problem (6.11) into two sub-problems corresponding to the forward and backward problems respectively. This split formulation is given by the following

$$\begin{cases} \min\limits_{\boldsymbol{u}} E_1^{IC}(\boldsymbol{u}) = \left\{ \int_\Omega |T_{\boldsymbol{u}} - R|^2 + \alpha \, |\nabla_3 \boldsymbol{u}|^2 + \beta \, |\boldsymbol{u} + \boldsymbol{v}|^2 \, d\Omega \right\}, \\ \\ \min\limits_{\boldsymbol{v}} E_2^{IC}(\boldsymbol{v}) = \left\{ \int_\Omega |R_{\boldsymbol{v}} - T|^2 + \alpha \, |\nabla_3 \boldsymbol{v}|^2 + \beta \, |\boldsymbol{v} + \boldsymbol{u}|^2 \, d\Omega \right\}. \end{cases} \tag{6.12}$$

It can be shown the split minimisation problem (6.12) yields the following EL equations

$$\begin{cases} -\alpha \Delta_3 u_m + F_m^{IC}(\boldsymbol{u}, \boldsymbol{v}) = 0, \\ -\alpha \Delta_3 v_m + G_m^{IC}(\boldsymbol{u}, \boldsymbol{v}) = 0 \end{cases} \tag{6.13}$$

with respective Neumann boundary conditions

$$\nabla_3 u_m \cdot \boldsymbol{n} = 0, \; \nabla_3 v_m \cdot \boldsymbol{n} = 0$$

and with the force terms

$$\begin{aligned} F_m^{IC}(\boldsymbol{u}, \boldsymbol{v}) &= \beta \big[u_m + v_m \big] + \partial_{u_m} T_{\boldsymbol{u}} \big[T_{\boldsymbol{u}} - R \big], \\ G_m^{IC}(\boldsymbol{u}, \boldsymbol{v}) &= \beta \big[v_m + u_m \big] + \partial_{v_m} R_{\boldsymbol{v}} \big[R_{\boldsymbol{v}} - T \big] \end{aligned} \tag{6.14}$$

for $m = 1, 2, 3$.

6.3 Discretisation and numerical methods for the proposed 3D models

Within the optimise-discretise approach for solving the minimisation problems (6.6) and (6.12), we seek to find numerical approximations to the EL equations (6.7) and (6.13) respectively. In order for us to do this we first discretise the 3D image domain Ω^h into a $n_1 \times n_1 \times n_3$ mesh, with respective intervals

$$\boldsymbol{h} = (h_1, h_1, h_3) = \left(\frac{1}{n_1 - 1}, \frac{1}{n_1 - 1}, \frac{1}{n_3 - 1} \right)$$

and then apply a FDM.

Remark 6.3.1. *Similar to the 2D cases in §4.2.1 and §5.2.2 where we used a square $n \times n$ mesh due to the fact medical image slices in general being square, this is also why we use a $n_1 \times n_1 \times n_3$ mesh in the 3D case. While each individual slice of the 3D image is typically square, the number of slices which make up the 3D image does not usually coincide with the dimension of the slices, and so we do not consider a cube mesh.*

Using this approach, in addition to a 3D lexicographic ordering system linking the

discrete point (i,j,k) to the global index

$$K = (k-2)(n_1-2)^2 + (j-2)(n_1-2) + (i-1)$$

for $i, j = 2, \ldots, n_1 - 1$ and $k = 2, \ldots, n_3 - 1$, results in the following discrete EL equations for the diffusion model

$$-\alpha\big(\Delta_3^h u_m^h\big)_K + \big(F_m^{Diff}(\boldsymbol{u}^h)\big)_K = 0 \tag{6.15}$$

and inverse consistent model

$$\begin{cases} -\alpha\big(\Delta_3^h u_m^h\big)_K + \big(F_m^{IC}(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_K = 0, \\ -\alpha\big(\Delta_3^h v_m^h\big)_K + \big(G_m^{IC}(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_K = 0 \end{cases} \tag{6.16}$$

with discrete force terms

$$\begin{aligned} \big(F_m^{Diff}(\boldsymbol{u}^h)\big)_K &= \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K \Big[\big(T_{\boldsymbol{u}}^h\big)_K - \big(R^h\big)_K\Big], \\ \big(F_m^{IC}(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_K &= \beta\Big[\big(u_m^h\big)_K + \big(v_m^h\big)_K\Big] + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K \Big[\big(T_{\boldsymbol{u}}^h\big)_K - \big(R^h\big)_K\Big], \\ \big(G_m^{IC}(\boldsymbol{u}^h, \boldsymbol{v}^h)\big)_K &= \beta\Big[\big(v_m^h\big)_K + \big(u_m^h\big)_K\Big] + \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_K \Big[\big(R_{\boldsymbol{v}}^h\big)_K - \big(T^h\big)_K\Big] \end{aligned} \tag{6.17}$$

respectively for $m = 1, 2, 3$ and $K = 1, \ldots, (n_1-2)^2(n_3-2)$. In addition we use the following central FD approximations to estimate any derivatives

$$\begin{aligned} \big(\Delta_3^h u_m^h\big)_K \approx{}& \frac{1}{h_1^2}\Big[\big(u_m^h\big)_{K-n_1} + \big(u_m^h\big)_{K-1} - 4\big(u_m^h\big)_K + \big(u_m^h\big)_{K+1} + \big(u_m^h\big)_{K+n_1}\Big] \\ &+ \frac{1}{h_3^2}\Big[\big(u_m^h\big)_{K-n_3} - 2\big(u_m^h\big)_K + \big(u_m^h\big)_{K+n_3}\Big], \end{aligned}$$

$$\begin{aligned} \big(\partial_{u_1}^h T_{\boldsymbol{u}}^h\big)_K &\approx \frac{1}{2h_1}\Big[\big(T_{\boldsymbol{u}}^h\big)_{K+1} - \big(T_{\boldsymbol{u}}^h\big)_{K-1}\Big], \\ \big(\partial_{u_2}^h T_{\boldsymbol{u}}^h\big)_K &\approx \frac{1}{2h_1}\Big[\big(T_{\boldsymbol{u}}^h\big)_{K+n_1} - \big(T_{\boldsymbol{u}}^h\big)_{K-n_1}\Big], \\ \big(\partial_{u_3}^h T_{\boldsymbol{u}}^h\big)_K &\approx \frac{1}{2h_3}\Big[\big(T_{\boldsymbol{u}}^h\big)_{K+n_3} - \big(T_{\boldsymbol{u}}^h\big)_{K-n_3}\Big] \end{aligned} \tag{6.18}$$

with similar approximations for the $\big(\Delta_3^h v_m^h\big)_K$ and $\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_K$ derivatives for $m = 1, 2, 3$.

6.3.1 Two pointwise smoothers for the 3D diffusion model

Now we present two smoother schemes for solving the discrete EL equations (6.15) associated with the 3D diffusion model.

Pointwise smoother 1 $\big(S_1^{Diff}\big)$**.** For the first pointwise smoother we consider a simple uncoupled smoother which uses each PDE in (6.15) to update each displacement

individually. Consider the following fixed point iteration scheme

$$-\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \big(F_m^{Diff}\big(\boldsymbol{u}^h\big)\big)_K^{(l+1)} = 0 \tag{6.19}$$

where

$$\begin{aligned}
\big(F_1^{Diff}\big(\boldsymbol{u}^h\big)\big)_K^{(l+1)} &= \big(\partial_{u_1}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T^h\big(x_1+u_1^{(l+1)},x_2+u_2^{(l)},x_3+u_3^{(l)}\big)\big)_K - \big(R^h\big)_K\Big],\\
\big(F_2^{Diff}\big(\boldsymbol{u}^h\big)\big)_K^{(l+1)} &= \big(\partial_{u_2}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T^h\big(x_1+u_1^{(l)},x_2+u_2^{(l+1)},x_3+u_3^{(l)}\big)\big)_K - \big(R^h\big)_K\Big],\\
\big(F_3^{Diff}\big(\boldsymbol{u}^h\big)\big)_K^{(l+1)} &= \big(\partial_{u_3}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T^h\big(x_1+u_1^{(l)},x_2+u_2^{(l)},x_3+u_3^{(l+1)}\big)\big)_K - \big(R^h\big)_K\Big].
\end{aligned} \tag{6.20}$$

Next we linearise the non-linear arguments of the T^h terms using the same treatment which was used in [33], in other words we use first order Taylor expansions. After the linearisation, we are left with the following

$$\begin{aligned}
-\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_K^{(l)} + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(u_m^h\big)_K^{(l+1)} - \big(u_m^h\big)_K^{(l)}\Big]\\
-\big(R^h\big)_K\Big] = 0
\end{aligned} \tag{6.21}$$

for $m = 1, 2, 3$. Then to compute the $(l+1)$ updates in (6.21), we use a lexicographic SOR based method, in other words we compute

$$\big(u_m^h\big)_K^{(l+1)} = \big(1-\omega\big)\big(u_m^h\big)_K^{(l)} + \omega\big(\bar{u}_m^h\big)_K^{(l)} \tag{6.22}$$

where the update $\big(\bar{u}_m^h\big)_K^{(l)}$ is obtained using a GSLEX step and $0 < \omega < 2$ is a weighting parameter to be determined.

Pointwise smoother 2 $\big(S_2^{Diff}\big)$. For the second pointwise smoother we consider a fully coupled scheme similar to the one shown in [33]. Now rather than using each PDE in (6.15) to update each displacement independently, we instead update all displacements simultaneously within each PDE. To do this we use the fixed point iteration scheme (6.19) with

$$\big(F_m^{Diff}\big(\boldsymbol{u}^h\big)\big)_K^{(l+1)} = \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_K^{(l+1)} - \big(R^h\big)_K\Big] \tag{6.23}$$

where

$$\big(T_{\boldsymbol{u}}^h\big)_K^{(l+1)} \equiv \big(T^h\big(x_1+u_1^{(l+1)},x_2+u_2^{(l+1)},x_3+u_3^{(l+1)}\big)\big)_K.$$

Again we use first order Taylor expansions to linearise the force terms in (6.23), which then results in the following

$$\begin{aligned}
-\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(T_{\boldsymbol{u}}^h\big)_K^{(l)} + \sum_{s=1}^{3}\big(\partial_{u_s}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\Big[\big(u_s^h\big)_K^{(l+1)} - \big(u_s^h\big)_K^{(l)}\Big]\\
-\big(R^h\big)_K\Big] = 0
\end{aligned} \tag{6.24}$$

for which we also use a SOR method to compute the $(l+1)$ updates in (6.24).

6.3.2 Two pointwise smoother for the 3D inverse consistent model

Now we present two smoother schemes for solving the discrete EL equations (6.16) associated with the 3D inverse consistent model.

Pointwise smoother 1 $\left(S_1^{IC}\right)$**.** The first smoother we propose to use for the inverse consistent model is based upon the full coupling idea seen in smoother S_2^{Diff} for the diffusion model. For this we use the following fixed point iteration scheme

$$\begin{cases} -\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \big(F_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K^{(l+1)} = 0, \\ -\alpha\big(\Delta_3^h v_m^h\big)_K^{(l+1)} + \big(G_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K^{(l+1)} = 0 \end{cases} \tag{6.25}$$

where we have the following force terms

$$\begin{aligned} \big(F_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K^{(l+1)} &= \beta\left[\big(u_m^h\big)_K^{(l+1)} + \big(v_m^h\big)_K^{(l+1)}\right] + \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\left[\big(T_{\boldsymbol{u}}^h\big)_K^{(l)} - \big(R^h\big)_K\right] \\ \big(G_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K^{(l+1)} &= \beta\left[\big(v_m^h\big)_K^{(l+1)} + \big(u_m^h\big)_K^{(l+1)}\right] + \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_K^{(l)}\left[\big(R_{\boldsymbol{v}}^h\big)_K^{(l)} - \big(T^h\big)_K\right]. \end{aligned} \tag{6.26}$$

After linearising the force terms (6.26) using Taylor expansions, and substituting back into the fixed point schemes (6.25), we get

$$\begin{cases} -\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \beta\left[\big(u_m^h\big)_K^{(l+1)} + \big(v_m^h\big)_K^{(l+1)}\right] \\ +\big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\left[\big(T_{\boldsymbol{u}}^h\big)_K^{(l)} + \sum_{s=1}^{3}\big(\partial_{u_s}^h T_{\boldsymbol{u}}^h\big)_K^{(l)}\left[\big(u_s^h\big)_K^{(l+1)} - \big(u_s^h\big)_K^{(l)}\right] - \big(R^h\big)_K\right] = 0 \\ \\ -\alpha\big(\Delta_3^h v_m^h\big)_K^{(l+1)} + \beta\left[\big(v_m^h\big)_K^{(l+1)} + \big(u_m^h\big)_K^{(l+1)}\right] \\ +\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)_K^{(l)}\left[\big(R_{\boldsymbol{v}}^h\big)_K^{(l)} + \sum_{s=1}^{3}\big(\partial_{v_s}^h R_{\boldsymbol{v}}^h\big)_K^{(l)}\left[\big(v_s^h\big)_K^{(l+1)} - \big(v_s^h\big)_K^{(l)}\right] - \big(T^h\big)_K\right] = 0 \end{cases} \tag{6.27}$$

and we compute the $(l+1)$ updates in (6.27) using an SOR method of the form

$$\big(\boldsymbol{w}^h\big)_K^{(l+1)} = \big(1-\omega\big)\big(\boldsymbol{w}^h\big)_K^{(l)} + \omega\big(\bar{\boldsymbol{w}}^h\big)_K^{(l)} \tag{6.28}$$

where

$$\boldsymbol{w}^h = \big[\boldsymbol{u}^h, \boldsymbol{v}^h\big]^T.$$

Pointwise smoother 2 (S_2^{IC})**.** Now in order to compute the $\big(\boldsymbol{w}^h\big)_K^{(l+1)}$ terms in (6.28), we must solve a 6×6 matrix equation for every discrete point K. Since the system matrix does not possess many zero entries, the inversion of this matrix (necessary for solving the system) can potentially be quite expensive computationally since there is no clear structure we can exploit. For this reason we propose a simplified version of

smoother S_1^{IC} which is based on the partially coupled smoother proposed in §5.2.4. To implement the partially coupled smoother, we use the fixed point scheme (6.25) with force terms now given by

$$\begin{aligned}
\big(F_1^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(u_1^h\big)_K^{(l+1)} + \big(v_1^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{u_1} T^h_{\boldsymbol{u}}\big)_K^{(l)}\left[\big(T^h\big(x_1+u_1^{(l+1)},x_2+u_2^{(l)},x_3+u_3^{(l)}\big)\big)_K - \big(R^h\big)_K\right] \\
\big(F_2^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(u_2^h\big)_K^{(l+1)} + \big(v_2^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{u_2} T^h_{\boldsymbol{u}}\big)_K^{(l)}\left[\big(T^h\big(x_1+u_1^{(l)},x_2+u_2^{(l+1)},x_3+u_3^{(l)}\big)\big)_K - \big(R^h\big)_K\right] \\
\big(F_3^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(u_3^h\big)_K^{(l+1)} + \big(v_3^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{u_3} T^h_{\boldsymbol{u}}\big)_K^{(l)}\left[\big(T^h\big(x_1+u_1^{(l)},x_2+u_2^{(l)},x_3+u_3^{(l+1)}\big)\big)_K - \big(R^h\big)_K\right]
\end{aligned}$$

$$\begin{aligned}
\big(G_1^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(v_1^h\big)_K^{(l+1)} + \big(u_1^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{v_1} R^h_{\boldsymbol{v}}\big)_K^{(l)}\left[\big(R^h\big(x_1+v_1^{(l+1)},x_2+v_2^{(l)},x_3+v_3^{(l)}\big)\big)_K - \big(T^h\big)_K\right] \\
\big(G_2^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(v_2^h\big)_K^{(l+1)} + \big(u_2^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{v_2} R^h_{\boldsymbol{v}}\big)_K^{(l)}\left[\big(R^h\big(x_1+v_1^{(l)},x_2+v_2^{(l+1)},x_3+v_3^{(l)}\big)\big)_K - \big(T^h\big)_K\right] \\
\big(G_3^{IC}(\boldsymbol{u}^h,\boldsymbol{v}^h)\big)_K^{(l+1)} &= \beta\left[\big(v_3^h\big)_K^{(l+1)} + \big(u_3^h\big)_K^{(l+1)}\right] \\
&\quad + \big(\partial^h_{v_3} R^h_{\boldsymbol{v}}\big)_K^{(l)}\left[\big(R^h\big(x_1+v_1^{(l)},x_2+v_2^{(l)},x_3+v_3^{(l+1)}\big)\big)_K - \big(T^h\big)_K\right]
\end{aligned} \tag{6.29}$$

which, after using Taylor approximations and substituting back into (6.25), gives

$$\begin{cases}
-\alpha\big(\Delta_3^h u_m^h\big)_K^{(l+1)} + \beta\left[\big(u_m^h\big)_K^{(l+1)} + \big(v_m^h\big)_K^{(l+1)}\right] \\
+\big(\partial^h_{u_m} T^h_{\boldsymbol{u}}\big)_K^{(l)}\left[\big(T^h_{\boldsymbol{u}}\big)_K^{(l)} + \big(\partial^h_{u_m} T^h_{\boldsymbol{u}}\big)_K^{(l)}\left[\big(u_m^h\big)_K^{(l+1)} - \big(u_m^h\big)_K^{(l)}\right] - \big(R^h\big)_K\right] = 0 \\
\\
-\alpha\big(\Delta_3^h v_m^h\big)_K^{(l+1)} + \beta\left[\big(v_m^h\big)_K^{(l+1)} + \big(u_m^h\big)_K^{(l+1)}\right] \\
+\big(\partial^h_{v_m} R^h_{\boldsymbol{v}}\big)_K^{(l)}\left[\big(R^h_{\boldsymbol{v}}\big)_K^{(l)} + \big(\partial^h_{v_m} R^h_{\boldsymbol{v}}\big)_K^{(l)}\left[\big(v_m^h\big)_K^{(l+1)} - \big(v_m^h\big)_K^{(l)}\right] - \big(T^h\big)_K\right] = 0.
\end{cases} \tag{6.30}$$

Again we use a SOR based method to determine the $(l+1)$ updates in (6.30).

6.4 3D non-linear multigrid and analysis for proposed models

As we mentioned in Chapters 4 and 5, the discrete EL equations can be very expensive computationally to solve, and for this reason we sought to implement a fast NMG framework to reduce the computational cost. In the 3D case this problem with computational cost is even more severe, and so the implementation of a fast NMG becomes a necessity. We now introduce our proposed NMG method, based upon those presented

in §4.2.3, §4.4.4 and §5.2.3, before performing the necessary analysis to obtain optimal convergence.

6.4.1 The 3D NMG framework

From previous chapters we stated MG methods result from two important ideas, i.e. iterative methods like the Gauss-Seidel method perform well as smoother schemes removing high frequency error components and smooth errors can be well approximated on coarse grids. To begin, let us denote the fine grid by Ω^h (with spacing $\boldsymbol{h} = \big(h_1, h_1, h_3\big)$) and the coarse grid by Ω^H (with spacing $\boldsymbol{H} = 2\boldsymbol{h}$) in the two-grid setting. Also let us write the discrete PDEs (6.15) and (6.16) using the following operator notation

$$\boldsymbol{\mathcal{N}}_h^{Diff}\big[\boldsymbol{u}^h\big] = \boldsymbol{\mathcal{G}}_h^{Diff} \tag{6.31}$$

and

$$\begin{cases} \bar{\boldsymbol{\mathcal{N}}}_h^{IC}\left[\boldsymbol{u}^h, \boldsymbol{v}^h\right] = \bar{\boldsymbol{\mathcal{G}}}_h^{IC}, \\ \tilde{\boldsymbol{\mathcal{N}}}_h^{IC}\left[\boldsymbol{u}^h, \boldsymbol{v}^h\right] = \tilde{\boldsymbol{\mathcal{G}}}_h^{IC} \end{cases} \tag{6.32}$$

respectively with

$$\boldsymbol{\mathcal{N}}_h^{Diff}\big[\boldsymbol{u}^h\big] = \begin{bmatrix} \big(\mathcal{N}_{1\,h}^{Diff}\big)_K \\ \big(\mathcal{N}_{2\,h}^{Diff}\big)_K \\ \big(\mathcal{N}_{3\,h}^{Diff}\big)_K \end{bmatrix},$$

$$\bar{\boldsymbol{\mathcal{N}}}_h^{IC}\big[\boldsymbol{u}^h, \boldsymbol{v}^h\big] = \begin{bmatrix} \big(\bar{\mathcal{N}}_{1\,h}^{IC}\big)_K \\ \big(\bar{\mathcal{N}}_{2\,h}^{IC}\big)_K \\ \big(\bar{\mathcal{N}}_{3\,h}^{IC}\big)_K \end{bmatrix}, \tilde{\boldsymbol{\mathcal{N}}}_h^{IC}\big[\boldsymbol{u}^h, \boldsymbol{v}^h\big] = \begin{bmatrix} \big(\tilde{\mathcal{N}}_{1\,h}^{IC}\big)_K \\ \big(\tilde{\mathcal{N}}_{2\,h}^{IC}\big)_K \\ \big(\tilde{\mathcal{N}}_{3\,h}^{IC}\big)_K \end{bmatrix},$$

$$\boldsymbol{\mathcal{G}}_h^{Diff} = \begin{bmatrix} \big(\mathcal{G}_{1\,h}^{Diff}\big)_K \\ \big(\mathcal{G}_{2\,h}^{Diff}\big)_K \\ \big(\mathcal{G}_{3\,h}^{Diff}\big)_K \end{bmatrix}, \bar{\boldsymbol{\mathcal{G}}}_h^{IC} = \begin{bmatrix} \big(\bar{\mathcal{G}}_{1\,h}^{IC}\big)_K \\ \big(\bar{\mathcal{G}}_{2\,h}^{IC}\big)_K \\ \big(\bar{\mathcal{G}}_{3\,h}^{IC}\big)_K \end{bmatrix}, \tilde{\boldsymbol{\mathcal{G}}}_h^{IC} = \begin{bmatrix} \big(\tilde{\mathcal{G}}_{1\,h}^{IC}\big)_K \\ \big(\tilde{\mathcal{G}}_{2\,h}^{IC}\big)_K \\ \big(\tilde{\mathcal{G}}_{3\,h}^{IC}\big)_K \end{bmatrix} \tag{6.33}$$

and

$$\big(\mathcal{N}_{m\,h}^{Diff}\big)_K = -\alpha\big(\Delta_3^h u_m^h\big)_K + \big(F_m^{Diff}\big(\boldsymbol{u}^h\big)\big)_K,$$

$$\begin{aligned} \big(\bar{\mathcal{N}}_{m\,h}^{IC}\big)_K &= -\alpha\big(\Delta_3^h u_m^h\big)_K + \big(F_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K = 0, \\ \big(\tilde{\mathcal{N}}_{m\,h}^{IC}\big)_K &= -\alpha\big(\Delta_3^h v_m^h\big)_K + \big(G_m^{IC}\big(\boldsymbol{u}^h, \boldsymbol{v}^h\big)\big)_K = 0, \end{aligned}$$

$$\big(\bar{\mathcal{G}}_{m\,h}^{IC}\big)_K = \big(\bar{\mathcal{G}}_{m\,h}^{IC}\big)_K = \big(\tilde{\mathcal{G}}_{m\,h}^{IC}\big)_K = 0$$

for $m = 1, 2, 3$ and $K = 1, \ldots, (n_1 - 2)^2(n_3 - 2)$. Then the proposed 3D FAS-NMG framework, in the two grid setting, for the diffusion, constrained diffusion and inverse consistent models can be seen in Algorithm 6, Algorithm 12 and Algorithm 13 respectively.

Before we can use the proposed smoothers we outlined in §6.3, we must first determine whether they are suitable for use in the NMG framework, in addition to how effective they are at smoothing errors. To determine these properties we use LFA to compute the h-ellipticity and smoothing rates for each of the proposed smoothers.

6.4.2 H-ellipticity for the proposed smoothers

By computing the h-ellipticity of a given smoother scheme, we can determine whether the scheme can be used as a pointwise error smoothing procedure within a NMG framework. We now demonstrate the proposed smoothers from §6.3.1 and §6.3.2 are suitable to be used as error smoothing procedures in our proposed NMG framework.

H-ellipticity for smoother S_1^{Diff}. Let us begin by writing the linearised system of PDEs in the following operator form

$$\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}\boldsymbol{u}^h = \boldsymbol{\mathcal{G}}_{1\,h}^{Diff} \tag{6.34}$$

where

$$\begin{aligned}
\boldsymbol{\mathcal{L}}_{1\,h}^{Diff} &= \begin{bmatrix} -\alpha\Delta_3^h + \sigma_{11}^h & 0 & 0 \\ 0 & -\alpha\Delta_3^h + \sigma_{22}^h & 0 \\ 0 & 0 & -\alpha\Delta_3^h + \sigma_{33}^h \end{bmatrix}, \boldsymbol{u}^h = \begin{bmatrix} u_1^h \\ u_2^h \\ u_3^h \end{bmatrix}, \\
\boldsymbol{\mathcal{G}}_{1\,h}^{Diff} &= \begin{bmatrix} g_1^h - F_1^{Diff}(\boldsymbol{u}^h) \\ g_2^h - F_2^{Diff}(\boldsymbol{u}^h) \\ g_3^h - F_3^{Diff}(\boldsymbol{u}^h) \end{bmatrix}
\end{aligned} \tag{6.35}$$

with

$$\begin{aligned}
F_m^{Diff}(\boldsymbol{u}^h) &= (\partial_{u_m}^h T_{\boldsymbol{u}}^h)^2 u_m^h - (\partial_{u_m}^h T_{\boldsymbol{u}}^h)[T_{\boldsymbol{u}}^h - R^h], \\
\sigma_{pq}^h &= (\partial_{u_m}^h T_{\boldsymbol{u}}^h)(\partial_{u_q}^h T_{\boldsymbol{u}}^h),\ g_m^h = 0
\end{aligned} \tag{6.36}$$

for m, p, $q = 1, 2, 3$. Applying the discrete linear operator $\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}$, to the grid functions $\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta})$ gives

$$\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{L}}}_{1\,h}^{Diff}(\boldsymbol{\theta})\boldsymbol{\Phi}^h(\boldsymbol{x}, \boldsymbol{\theta}) \tag{6.37}$$

with Fourier symbol

$$\hat{\boldsymbol{\mathcal{L}}}_{1\,h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) & 0 & 0 \\ 0 & \sigma_{22}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) & 0 \\ 0 & 0 & \sigma_{33}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) \end{bmatrix} \tag{6.38}$$

where $\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the 3D discrete Laplace operator Δ_3^h. We compute the h-ellipticity from the following

$$\mathscr{E}_{1\,h}^{Diff}(\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}) = \frac{\min\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\}}{\max\left\{\left|\det\left(\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\right)\right| : \boldsymbol{\theta} \in \boldsymbol{\Theta}\right\}} \tag{6.39}$$

where

$$\boldsymbol{\Theta} = [-\pi, \pi)^3,\ \boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^3.$$

It can be shown

$$\det\left(\hat{\boldsymbol{\mathcal{L}}}_1^h(\boldsymbol{\theta})\right) = -\alpha^3\left(\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})\right)^3 + \alpha^2 c_1^h\left(\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})\right)^2 - \alpha c_2^h\left(\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})\right) + c_3^h \tag{6.40}$$

where

$$c_1^h = \sigma_{11}^h + \sigma_{22}^h + \sigma_{33}^h,\ c_2^h = \sigma_{11}^h\sigma_{22}^h + \sigma_{11}^h\sigma_{33}^h + \sigma_{22}^h\sigma_{33}^h. \tag{6.41}$$

Using well known results, we can show

$$\begin{aligned} -\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) &= \frac{2}{h^2}\left[3 - \left(\cos\theta_1 + \cos\theta_2 + \cos\theta_3\right)\right], \\ \min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}_{high}}\left\{\left(-\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})\right)\right\} &= \frac{2}{h^2},\ \max_{\boldsymbol{\theta}\in\boldsymbol{\Theta}}\left\{\left(-\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})\right)\right\} = \frac{12}{h^2}. \end{aligned} \tag{6.42}$$

Substituting (6.42) and (6.40) back into (6.39), in addition to taking the limit as $h \to 0$, we get

$$\lim_{h\to 0}\left\{\mathscr{E}_{1\,h}^{Diff}(\boldsymbol{\mathcal{L}}_{1\,h}^{Diff})\right\} = \lim_{h\to 0}\left\{\frac{8\alpha^3 + \mathcal{O}(h)}{1728\alpha^3 + \mathcal{O}(h)}\right\} = \frac{1}{216}. \tag{6.43}$$

Since the h-ellipticity value (6.43) is bounded away from 0, as $h \to 0$, we conclude the smoother S_1^{Diff} is suitable for use as a pointwise error smoothing procedure.

H-ellipticity for smoother S_2^{Diff}. Using a similar procedure to the one we first showed for smoother S_1^{Diff}, we can obtain the following Fourier symbol for the discrete operator $\boldsymbol{\mathcal{L}}_{2\,h}^{Diff}$

$$\hat{\boldsymbol{\mathcal{L}}}_{2\,h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} \sigma_{11}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) & \sigma_{12}^h & \sigma_{13}^h \\ \sigma_{12}^h & \sigma_{22}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) & \sigma_{23}^h \\ \sigma_{13}^h & \sigma_{23}^h & \sigma_{33}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) \end{bmatrix} \tag{6.44}$$

and we compute the h-ellipticity using (6.39) with the Fourier symbol (6.44). Computing the h-ellipticity, and again taking the limit $h \to 0$, gives

$$\lim_{h\to 0}\left\{\mathscr{E}_{2\,h}^{Diff}\big(\boldsymbol{\mathcal{L}}_{2\,h}^{Diff}\big)\right\} = \lim_{h\to 0}\left\{\frac{8\alpha^3+\mathcal{O}(h)}{1728\alpha^3+\mathcal{O}(h)}\right\} = \frac{1}{216}. \tag{6.45}$$

and so we see the smoother S_2^{Diff} is also suitable as a pointwise error smoothing procedure.

H-ellipticity for smoother S_1^{IC}. Now we repeat the h-ellipticity calculation for the fully coupled smoother S_1^{IC} for the inverse consistent model. We begin in a similar way to the S_1^{Diff} smoother for the diffusion model, in other words we write the system (6.25) using the following operator form

$$\boldsymbol{\mathcal{L}}_{1\,h}^{IC}\boldsymbol{w}^h = \boldsymbol{\mathcal{G}}_{1\,h}^{IC} \tag{6.46}$$

where

$$\boldsymbol{\mathcal{L}}_{1\,h}^{IC} = \begin{bmatrix} a_{11}^h-\alpha\Delta_3^h & \sigma_{12}^h & \sigma_{13}^h & \beta & 0 & 0\\ \sigma_{12}^h & a_{22}^h-\alpha\Delta_3^h & \sigma_{23}^h & 0 & \beta & 0\\ \sigma_{13}^h & \sigma_{23}^h & a_{33}^h-\alpha\Delta_3^h & 0 & 0 & \beta\\ \beta & 0 & 0 & b_{11}^h-\alpha\Delta_3^h & \tau_{12}^h & \tau_{13}^h\\ 0 & \beta & 0 & \tau_{12}^h & b_{22}^h-\alpha\Delta_3^h & \tau_{23}^h\\ 0 & 0 & \beta & \tau_{13}^h & \tau_{23}^h & b_{33}^h-\alpha\Delta_3^h \end{bmatrix},$$

$$\boldsymbol{w}^h = \begin{bmatrix} \boldsymbol{u}_1^h\\ \boldsymbol{u}_2^h\\ \boldsymbol{u}_3^h\\ \boldsymbol{v}_1^h\\ \boldsymbol{v}_2^h\\ \boldsymbol{v}_3^h \end{bmatrix},\ \boldsymbol{\mathcal{G}}_{1\,h}^{IC} = \begin{bmatrix} g_1^h - F_{1\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big)\\ g_2^h - F_{2\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big)\\ g_3^h - F_{3\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big)\\ g_4^h - G_{1\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big)\\ g_5^h - G_{2\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big)\\ g_6^h - G_{3\,h}^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) \end{bmatrix} \tag{6.47}$$

also where

$$\begin{aligned} F_m^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) &= \sum_{s=1}^{3}\big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)\big(\partial_{u_s}^h T_{\boldsymbol{u}}^h\big)u_s^h - \big(\partial_{u_m}^h T_{\boldsymbol{u}}^h\big)\big[T_{\boldsymbol{u}}^h - R^h\big],\\ G_m^{IC}\big(\boldsymbol{u}^h,\boldsymbol{v}^h\big) &= \sum_{s=1}^{3}\big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)\big(\partial_{v_s}^h R_{\boldsymbol{v}}^h\big)v_s^h - \big(\partial_{v_m}^h R_{\boldsymbol{v}}^h\big)\big[R_{\boldsymbol{v}}^h - T^h\big],\\ \sigma_{pq}^h &= \big(\partial_{u_p}^h T_{\boldsymbol{u}}^h\big)\big(\partial_{u_q}^h T_{\boldsymbol{u}}^h\big),\ \tau_{pq}^h = \big(\partial_{v_p}^h R_{\boldsymbol{v}}^h\big)\big(\partial_{v_q}^h R_{\boldsymbol{v}}^h\big),\\ a_{pp}^h &= \sigma_{pp}^h + \beta,\ b_{pp}^h = \tau_{pp}^h + \beta \end{aligned} \tag{6.48}$$

for m, p, $q = 1, 2, 3$. Following the same step as for the diffusion model smoothers, we apply the discrete operator $\boldsymbol{\mathcal{L}}_{1\,h}^{IC}$ to the grid functions $\boldsymbol{\Phi}^h\big(\boldsymbol{x},\boldsymbol{\theta}\big)$ to get

$$\boldsymbol{\mathcal{L}}_{1\,h}^{IC}\boldsymbol{\Phi}\big(\boldsymbol{x},\boldsymbol{\theta}\big) = \hat{\boldsymbol{\mathcal{L}}}_{1\,h}^{IC}\big(\boldsymbol{\theta}\big)\boldsymbol{\Phi}^h\big(\boldsymbol{x},\boldsymbol{\theta}\big) \tag{6.49}$$

with Fourier symbol

$$\hat{\boldsymbol{\mathcal{L}}}_{1\,h}^{IC}(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}_{11}^h & \sigma_{11}^h & \sigma_{13}^h & \beta & 0 & 0 \\ \sigma_{12}^h & \tilde{a}_{22}^h & \sigma_{23}^h & 0 & \beta & 0 \\ \sigma_{13}^h & \sigma_{23}^h & \tilde{a}_{33}^h & 0 & 0 & \beta \\ \beta & 0 & 0 & \tilde{b}_{11}^h & \tau_{12}^h & \tau_{13}^h \\ 0 & \beta & 0 & \tau_{12}^h & \tilde{b}_{22}^h & \tau_{13}^h \\ 0 & 0 & \beta & \tau_{13}^h & \tau_{23}^h & \tilde{b}_{33}^h \end{bmatrix} \tag{6.50}$$

where

$$\tilde{a}_{mm}^h = a_{mm}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}),\ \tilde{b}_{mm}^h = b_{mm}^h - \alpha\hat{\mathscr{L}}_3^h(\boldsymbol{\theta}) \tag{6.51}$$

and $\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the 3D discrete Laplace operator for $m = 1, 2, 3$. Again we compute the h-ellipticity using (6.39), except now with the Fourier symbol (6.50), along with the results (6.42). Then it can be shown the h-ellipticity for smoother S_1^{IC}, after taking the limit as $h \to 0$, is given by

$$\lim_{h\to 0}\left\{\mathscr{E}_{1\,h}^{IC}(\boldsymbol{\mathcal{L}}_{1\,h}^{IC})\right\} = \lim_{h\to 0}\left\{\frac{64\alpha^6 + \mathcal{O}(h)}{2985984\alpha^6 + \mathcal{O}(h)}\right\} = \frac{1}{46656}. \tag{6.52}$$

Thus we reach the same conclusion we did for the diffusion model smoothers, namely the smoother S_1^{IC} is suitable as a pointwise error smoothing procedure since the h-ellipticity is always bounded away from 0.

H-ellipticity for smoother S_2^{IC}. Finally we perform the h-ellipticity calculation for smoother S_2^{IC}. It can be shown we obtain the following Fourier symbol for the discrete operator $\boldsymbol{\mathcal{L}}_{2\,h}^{IC}$

$$\hat{\boldsymbol{\mathcal{L}}}_{2\,h}^{IC}(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}_{11}^h & 0 & 0 & \beta & 0 & 0 \\ 0 & \tilde{a}_{22}^h & 0 & 0 & \beta & 0 \\ 0 & 0 & \tilde{a}_{33}^h & 0 & 0 & \beta \\ \beta & 0 & 0 & \tilde{b}_{11}^h & 0 & 0 \\ 0 & \beta & 0 & 0 & \tilde{b}_{22}^h & 0 \\ 0 & 0 & \beta & 0 & 0 & \tilde{b}_{33}^h \end{bmatrix}. \tag{6.53}$$

again where $\tilde{a}_{mm}^h$, $\tilde{b}_{mm}^h$ are as shown in (6.51) and $\hat{\mathscr{L}}_3^h(\boldsymbol{\theta})$ denotes the Fourier symbol of the 3D discrete Laplace operator for $m = 1, 2, 3$. Then using (6.39), along with (6.42) and (6.53), we obtain the same h-ellipticity value as seen in (6.52). Therefore our partially coupled smoother S_2^{IC} is also suitable for use as a pointwise error smoothing procedure.

6.4.3 Smoothing rate analysis of the proposed smoothers

Now we consider how effective the proposed smoother schemes are at smoothing the high frequency error components. Using this analysis we obtain a guide for the number

of smoothing steps which will be required to smooth the error to a tolerance of 10^{-1} (typical for multigrid schemes). Moreover, we also use this analysis to see how the value of the parameter ω in the SOR scheme effects the smoothing rate, thus allowing us to select an optimal value for each smoother to achieve the best smoothing rates. In order to estimate the smoothing rates we again use LFA.

Smoothing rate analysis for smoother S_1^{Diff}. To begin let us write the discrete system (6.19) in the following way

$$\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}\boldsymbol{u}^h + \boldsymbol{\mathcal{M}}_{1\,h}^{Diff}\boldsymbol{u}^h = \boldsymbol{\mathcal{G}}_{1\,h}^{Diff} \tag{6.54}$$

where $\boldsymbol{\mathcal{L}}_{1\,h}^{Diff}$, $\boldsymbol{u}^h$, $\boldsymbol{\mathcal{G}}_{1\,h}^{Diff}$ are as in (6.35), and

$$\boldsymbol{\mathcal{M}}_{1\,h}^{Diff} = \begin{bmatrix} -\sigma_{11}^h & 0 & 0 \\ 0 & -\sigma_{22}^h & 0 \\ 0 & 0 & -\sigma_{33}^h \end{bmatrix} \tag{6.55}$$

with σ_{pq}^h as in (6.36) for p, $q = 1, 2$. Moreover, let us use the splitting

$$\Delta_3^h = \mathscr{L}_{3+}^h + \mathscr{L}_{3-}^h$$

where $\mathscr{L}_{3+}^h$, $\mathscr{L}_{3-}^h$ are given by the following 3D stencils

$$\begin{aligned} \mathscr{L}_{3+}^h &= \frac{1}{h^2}\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 1 & -\frac{6}{\omega} & 0 \\ 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right), \\ \mathscr{L}_{3+}^h &= \frac{1}{h^2}\left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 1 & 0 \\ 0 & -6\big(1-\frac{1}{\omega}\big) & 1 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}\right). \end{aligned} \tag{6.56}$$

Using the stencils (6.56), we can write (6.54) in the following way

$$\boldsymbol{\mathcal{L}}_{1+h}^{Diff}\boldsymbol{u}_{new}^h + \boldsymbol{\mathcal{L}}_{1-h}^{Diff}\boldsymbol{u}_{old}^h + \boldsymbol{\mathcal{M}}_{1\,h}^{Diff}\boldsymbol{u}_{old}^h = \boldsymbol{\mathcal{G}}_{1\,h}^{Diff} \tag{6.57}$$

where $\boldsymbol{u}_{old}^h$, $\boldsymbol{u}_{new}^h$ denote the previous and current approximations of $\boldsymbol{u}^h$ respectively, and also where

$$\begin{aligned} \boldsymbol{\mathcal{L}}_{1+h}^{Diff} &= \begin{bmatrix} \sigma_{11}^h - \mathscr{L}_{3+}^h & 0 & 0 \\ 0 & \sigma_{22}^h - \mathscr{L}_{3+}^h & 0 \\ 0 & 0 & \sigma_{33}^h - \mathscr{L}_{3+}^h \end{bmatrix}, \\ \boldsymbol{\mathcal{L}}_{1-h}^{Diff} &= \begin{bmatrix} -\mathscr{L}_{3-}^h & 0 & 0 \\ 0 & -\mathscr{L}_{3-}^h & 0 \\ 0 & 0 & -\mathscr{L}_{3-}^h \end{bmatrix} \end{aligned} \tag{6.58}$$

with $\boldsymbol{\mathcal{M}}_{1\,h}^{Diff}$ as defined in (6.55). After subtracting (6.57) from (6.54), we obtain the local error equations given by the following

$$\left[\boldsymbol{\mathcal{L}}_{1+h}^{Diff}\right] \boldsymbol{e}_{new}^{h} = -\left[\boldsymbol{\mathcal{L}}_{1-h}^{Diff} + \boldsymbol{\mathcal{M}}_{1\,h}^{Diff}\right] \boldsymbol{e}_{old}^{h} \tag{6.59}$$

where

$$\boldsymbol{e}_{*}^{h} = \left[e_{1\,*}^{h}, e_{2\,*}^{h}, e_{3\,*}^{h}\right]^{T}.$$

Expanding the errors in (6.59) using Fourier components, we get the following

$$\boldsymbol{e}_{*}^{h} = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \boldsymbol{\psi}_{\boldsymbol{\theta}}^{*} \exp\left(\boldsymbol{i}\lambda_1 i + \boldsymbol{i}\lambda_2 j + \boldsymbol{i}\lambda_3 k\right) \tag{6.60}$$

where

$$\boldsymbol{i} = \sqrt{-1},\ \boldsymbol{\Theta} = [-\pi, \pi)^3,\ \lambda_m = \frac{2\theta_m \pi}{h}$$

and $\boldsymbol{\psi}_{\boldsymbol{\theta}}^{*}$ are Fourier coefficients for $m = 1, 2, 3$. Substituting (6.60) into (6.59) gives

$$\left[\hat{\boldsymbol{\mathcal{L}}}_{1+h}^{Diff}(\boldsymbol{\theta})\right] \boldsymbol{\psi}_{\boldsymbol{\theta}}^{new} \exp\left(\boldsymbol{i}\boldsymbol{\lambda}\cdot\boldsymbol{x}\right) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{1-h}^{Diff}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_{1\,h}^{Diff}(\boldsymbol{\theta})\right] \boldsymbol{\psi}_{\boldsymbol{\theta}}^{old} \exp\left(\boldsymbol{i}\boldsymbol{\lambda}\cdot\boldsymbol{x}\right)$$

where

$$\boldsymbol{\lambda} = \left(\lambda_1, \lambda_2, \lambda_3\right),\ \boldsymbol{x} = \left(i, j, k\right)$$

and

$$\hat{\boldsymbol{\mathcal{L}}}_{1+h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} a_{+}^{h} + \sigma_{11}^{h} & 0 & 0 \\ 0 & a_{+}^{h} + \sigma_{22}^{h} & 0 \\ 0 & 0 & a_{+}^{h} + \sigma_{33}^{h} \end{bmatrix},\ \hat{\boldsymbol{\mathcal{L}}}_{1-h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} a_{-}^{h} & 0 & 0 \\ 0 & a_{-}^{h} & 0 \\ 0 & 0 & a_{-}^{h} \end{bmatrix} \tag{6.61}$$

where $\hat{\boldsymbol{M}}_{1\,h}^{Diff}(\boldsymbol{\theta})$ is the same as $\boldsymbol{M}_{1\,h}^{Diff}$ from (6.58), and

$$a_{+}^{h} = \frac{\alpha}{h^2}\left[\frac{6}{\omega} - \left(e^{-\boldsymbol{i}\lambda_3} + e^{-\boldsymbol{i}\lambda_2} + e^{-\boldsymbol{i}\lambda_1}\right)\right],$$
$$a_{-}^{h} = \frac{\alpha}{h^2}\left[6\left(1 - \frac{1}{\omega}\right) - \left(e^{\boldsymbol{i}\lambda_3} + e^{\boldsymbol{i}\lambda_2} + e^{\boldsymbol{i}\lambda_1}\right)\right].$$

Then, the local smoothing rate is computed from the following

$$\mu_{loc} \equiv \mu_{loc}(\boldsymbol{\theta}) = \sup\left\{\rho\left(\hat{\boldsymbol{\mathcal{S}}}_{1\,h}^{Diff}(\boldsymbol{\theta})\right) : \boldsymbol{\theta} \in \boldsymbol{\Theta}_{high}\right\} \tag{6.62}$$

where

$$\boldsymbol{\Theta}_{high} = \boldsymbol{\Theta} \setminus \left[-\frac{\pi}{2}, \frac{\pi}{2}\right)^3$$

denotes the high frequency range, $\rho(\cdot)$ denotes the spectral radius and $\hat{\boldsymbol{\mathcal{S}}}_{1\,h}^{Diff}(\boldsymbol{\theta})$ is the amplification matrix given by

$$\hat{\boldsymbol{\mathcal{S}}}_{1\,h}^{Diff}(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{1+h}^{Diff}(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}_{1-h}^{Diff}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_{1\,h}^{Diff}(\boldsymbol{\theta})\right]. \tag{6.63}$$

Smoothing rate analysis for smoother S_2^{Diff}. Repeating the smoothing rate calculation for smoother S_2^{Diff}, we compute the smoothing rate using (6.62) with amplification matrix

$$\hat{\boldsymbol{\mathcal{S}}}_{2\,h}^{Diff}(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{2+h}^{Diff}(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}_{2-h}^{Diff}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_{2\,h}^{Diff}(\boldsymbol{\theta})\right]. \tag{6.64}$$

where $\hat{\boldsymbol{\mathcal{L}}}_{2-h}^{Diff}(\boldsymbol{\theta}) = \hat{\boldsymbol{\mathcal{L}}}_{1-h}^{Diff}(\boldsymbol{\theta})$ from (6.61), and

$$\hat{\boldsymbol{\mathcal{L}}}_{2+h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} a_+^h + \sigma_{11}^h & \sigma_{12}^h & \sigma_{13}^h \\ \sigma_{12}^h & a_+^h + \sigma_{22}^h & \sigma_{23}^h \\ \sigma_{13}^h & \sigma_{23}^h & a_+^h + \sigma_{33}^h \end{bmatrix},$$

$$\hat{\boldsymbol{\mathcal{M}}}_{2\,h}^{Diff}(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma_{11}^h & -\sigma_{12}^h & -\sigma_{13}^h \\ -\sigma_{12}^h & -\sigma_{22}^h & -\sigma_{23}^h \\ -\sigma_{13}^h & -\sigma_{23}^h & -\sigma_{33}^h \end{bmatrix}.$$

Smoothing rate analysis for smoother S_1^{IC}. Similar to the smoothing analysis we performed for smoother S_1^{Diff}, we begin by writing the discrete system (6.25) in the following way

$$\boldsymbol{\mathcal{L}}_{1\,h}^{IC}\boldsymbol{w}^h + \boldsymbol{\mathcal{M}}_{1\,h}^{IC}\boldsymbol{w}^h = \boldsymbol{\mathcal{G}}_{1\,h}^{IC} \tag{6.65}$$

where $\boldsymbol{\mathcal{L}}_{1\,h}^{IC}$, $\boldsymbol{w}^h$ and $\boldsymbol{\mathcal{G}}_{1\,h}^{IC}$ are defined in (6.47), also with

$$\boldsymbol{\mathcal{M}}_{1\,h}^{IC} = \begin{bmatrix} -\sigma_{11}^h & -\sigma_{12}^h & -\sigma_{13}^h & 0 & 0 & 0 \\ -\sigma_{12}^h & -\sigma_{22}^h & -\sigma_{23}^h & 0 & 0 & 0 \\ -\sigma_{13}^h & -\sigma_{23}^h & -\sigma_{33}^h & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau_{11}^h & -\tau_{12}^h & -\tau_{13}^h \\ 0 & 0 & 0 & -\tau_{12}^h & -\tau_{22}^h & -\tau_{23}^h \\ 0 & 0 & 0 & -\tau_{13}^h & -\tau_{23}^h & -\tau_{33}^h \end{bmatrix}. \tag{6.66}$$

Then following the same method shown for smoother S_1^{Diff}, we compute the local smoothing rate for smoother S_1^{IC} using (6.62) with the following amplification matrix

$$\hat{\boldsymbol{\mathcal{S}}}_{1\,h}^{IC}(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}_{1+h}^{IC}(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}_{1-h}^{IC}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}_{1\,h}^{IC}(\boldsymbol{\theta})\right] \tag{6.67}$$

where

$$\hat{\boldsymbol{\mathcal{L}}}^{IC}_{1+h}(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}^h_+ + \sigma^h_{11} & \sigma^h_{12} & \sigma^h_{13} & \beta & 0 & 0 \\ \sigma^h_{12} & \tilde{a}^h_+ + \sigma^h_{22} & \sigma^h_{23} & 0 & \beta & 0 \\ \sigma^h_{13} & \sigma^h_{23} & \tilde{a}^h_+ + \sigma^h_{33} & 0 & 0 & \beta \\ \beta & 0 & 0 & \tilde{a}^h_+ + \tau^h_{11} & \tau^h_{12} & \tau^h_{13} \\ 0 & \beta & 0 & \tau^h_{12} & \tilde{a}^h_+ + \tau^h_{22} & \tau^h_{13} \\ 0 & 0 & \beta & \tau^h_{13} & \tau^h_{23} & \tilde{a}^h_+ + \tau^h_{33} \end{bmatrix}$$

$$\hat{\boldsymbol{\mathcal{L}}}^{IC}_{1-h}(\boldsymbol{\theta}) = diag\big(\tilde{a}^h_-, \tilde{a}^h_-, \tilde{a}^h_-, \tilde{a}^h_-, \tilde{a}^h_-, \tilde{a}^h_-\big),$$

$$\hat{\boldsymbol{\mathcal{M}}}^{IC}_{1\,h}(\boldsymbol{\theta}) = \begin{bmatrix} -\sigma^h_{11} & -\sigma^h_{12} & -\sigma^h_{13} & 0 & 0 & 0 \\ -\sigma^h_{12} & -\sigma^h_{22} & -\sigma^h_{23} & 0 & 0 & 0 \\ -\sigma^h_{13} & -\sigma^h_{23} & -\sigma^h_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\tau^h_{11} & -\tau^h_{12} & -\tau^h_{13} \\ 0 & 0 & 0 & -\tau^h_{12} & -\tau^h_{22} & -\tau^h_{23} \\ 0 & 0 & 0 & -\tau^h_{13} & -\tau^h_{23} & -\tau^h_{33} \end{bmatrix} \tag{6.68}$$

and

$$\begin{aligned} \tilde{a}^h_+ &= \beta + \frac{\alpha}{h^2}\left[\frac{6}{\omega} - \left(e^{-\boldsymbol{i}\lambda_3} + e^{-\boldsymbol{i}\lambda_2} + e^{-\boldsymbol{i}\lambda_1}\right)\right], \\ \tilde{a}^h_- &= \beta + \frac{\alpha}{h^2}\left[6\left(1 - \frac{1}{\omega}\right) - \left(e^{\boldsymbol{i}\lambda_3} + e^{\boldsymbol{i}\lambda_2} + e^{\boldsymbol{i}\lambda_1}\right)\right] \end{aligned} \tag{6.69}$$

Smoothing rate analysis for smoother S^{IC}_2. Finally we perform the smoothing rate calculation for the smoother S^{IC}_2. We do this again using (6.62) with the following amplification matrix

$$\hat{\boldsymbol{\mathcal{S}}}^{IC}_{2\,h}(\boldsymbol{\theta}) = -\left[\hat{\boldsymbol{\mathcal{L}}}^{IC}_{2+h}(\boldsymbol{\theta})\right]^{-1}\left[\hat{\boldsymbol{\mathcal{L}}}^{IC}_{2-h}(\boldsymbol{\theta}) + \hat{\boldsymbol{\mathcal{M}}}^{IC}_{2\,h}(\boldsymbol{\theta})\right] \tag{6.70}$$

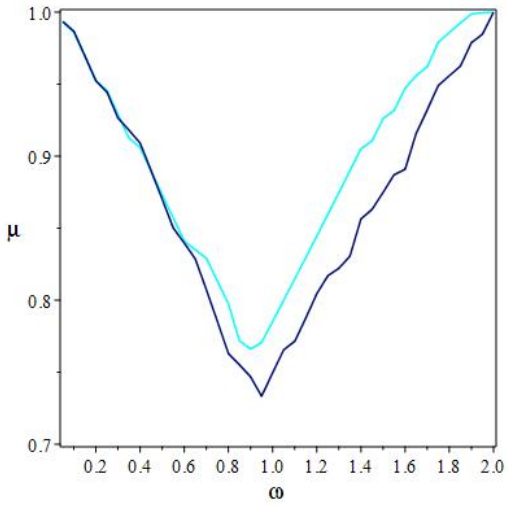
where $\hat{\boldsymbol{\mathcal{L}}}^{IC}_{2-h}(\boldsymbol{\theta})$ is the same as $\hat{\boldsymbol{\mathcal{L}}}^{IC}_{1-h}(\boldsymbol{\theta})$ from (6.68), and

$$\hat{\boldsymbol{\mathcal{L}}}^{IC}_{2+h}(\boldsymbol{\theta}) = \begin{bmatrix} \tilde{a}^h_+ + \sigma^h_{11} & 0 & 0 & \beta & 0 & 0 \\ 0 & \tilde{a}^h_+ + \sigma^h_{22} & 0 & 0 & \beta & 0 \\ 0 & 0 & \tilde{a}^h_+ + \sigma^h_{33} & 0 & 0 & \beta \\ \beta & 0 & 0 & \tilde{a}^h_+ + \tau^h_{11} & 0 & 0 \\ 0 & \beta & 0 & 0 & \tilde{a}^h_+ + \tau^h_{22} & 0 \\ 0 & 0 & \beta & 0 & 0 & \tilde{a}^h_+ + \tau^h_{33} \end{bmatrix}$$

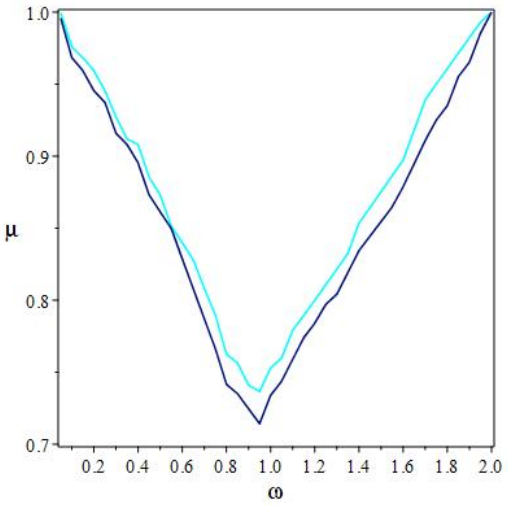
$$\hat{\boldsymbol{\mathcal{M}}}^{IC}_{2\,h}(\boldsymbol{\theta}) = diag\left[-\sigma^h_{11}, -\sigma^h_{22}, -\sigma^h_{33}, -\tau^h_{11}, -\tau^h_{22}, -\tau^h_{33}\right]$$

with $\tilde{a}^h_+$ as defined in (6.69).

Smoothing rate examples. In Figure 6.1 we have plotted the smoothing rates of



(a) Plot of the smoothing rate for the smoothers S_1^{Diff} and S_2^{Diff} for varying values of the parameter ω. Here the light blue line corresponds to the rates for smoother S_1^{Diff} while the dark blue line corresponds to the rates for smoother S_2^{Diff}.



(b) Plot of the smoothing rate for the smoothers S_1^{IC} and S_2^{IC} for varying values of the parameter ω. Here the dark blue line corresponds to the rates for smoother S_1^{IC} while the light blue line corresponds to the rates for smoother S_2^{IC}.

Figure 6.1: Illustrations of how the parameter ω affects the smoothing rates of the proposed smoothers for the diffusion and inverse consistent models.

S_1^{Diff}		S_2^{Diff}		S_1^{IC}		S_2^{IC}	
μ_{avg}	$Tol\ 10^{-1}$	μ_{avg}	$Tol\ 10^{-1}$	μ_{avg}	$Tol\ 10^{-1}$	μ_{avg}	$Tol\ 10^{-1}$
0.76118	9	0.73735	8	0.71424	7	0.72660	8

Table 6.1: Comparison of the smoothing rates of the proposed smoothers S_1^{Diff}, S_2^{Diff}, S_1^{IC} and S_2^{IC} for parameters $\alpha = \frac{1}{20}$ and $\beta = 10^4$ after five inner and outer iterations on a $32 \times 32 \times 32$ grid. For each smoother, the smoothing rates and number of inner iterations required to reach an error reduction of 10^{-1} are shown for rates corresponding to the optimal parameter ω according to Figure 6.1.

the four proposed smoothers S_1^{Diff}, S_2^{Diff}, S_1^{IC} and S_2^{IC} corresponding to forty values of the parameter ω in the range $(0, 2]$. Using these plots we can select the parameter ω which gives the best smoothing rate, and using this rate we can estimate the number of smoothing steps required to reach a tolerance in error reduction of 10^{-1} which is deemed to be sufficient for a NMG scheme. In all four cases we see the optimal values of ω are close to, but less than, 1. For smoothers S_1^{Diff}, S_1^{IC} and S_1^{IC} the parameter $\omega = 0.95$ is selected while for smoother S_2^{Diff} we use the parameter $\omega = 0.9$. Using these parameter values, we compute the number of smoothing steps l, for a smoother with smoothing rate $\mathcal{P}$, required to reach a tolerance of 10^{-1} according to

$$l = -\frac{\ln(10)}{\ln(\mathcal{P})}. \tag{6.71}$$

The results of the calculation of (6.71) for each of the four smoothers can be seen in Table 6.1. For the smoothers S_1^{Diff} and S_2^{Diff} associated with the diffusion models, we see the smoothing rate for smoother S_2^{Diff} is noticeably smaller than the rate for smoother S_1^{Diff}. Therefore for the diffusion and constrained diffusion models (as shown in §6.2.1 and §6.2.2 respectively), we use the NMG method as shown in Algorithm 6 and Algorithm 12 with smoother S_2^{Diff} and $\nu_1 = \nu_2 = 8$ which we denote by **DNMG3D** and **CDNMG3D** respectively.

Now for the smoothers S_1^{IC} and S_2^{IC} associated with the proposed inverse consistent model, from Table 6.1 we see the smoothing rate of smoother S_1^{IC} is marginally better than the rate for smoother S_2^{IC}. While this would normally imply smoother S_1^{IC} should be used in the NMG method instead of smoother S_2^{IC}, due to the fully coupled nature of the discrete PDEs shown in (6.27) we are required to solve a 6×6 inverse problem at every discrete voxel involving a dense system matrix which possesses no exploitable properties which allows us to reduce the cost of directly inverting the matrix. As a result the smoother S_1^{IC} is very expensive computationally, a problem which is compounded further when working in 3D. For smoother S_2^{IC} however, the semi-coupled equations shown in (6.30) allow us to reduce the inverse problem to simple scalar multiplication owing to the sparse structure of the system matrix thus greatly reducing the computational cost when compared with smoother S_1^{IC}. Hence for our proposed 3D inverse consistent model, we use the NMG method given by Algorithm 13 with smoother S_2^{IC} and $\nu_1 = \nu_2 = 8$ which we denote by **ICNMG3D**.

6.5 Preliminary numerical results

In this section we present some preliminary experimental results comparing registration accuracy of four different algorithms, these are:

(i) A standard 3D diffusion model, combined with the NMG scheme outlined in Algorithm 6, which we have denoted **DNMG3D**;

(ii) Our proposed 3D constrained diffusion model, with the NMG scheme outlined in Algorithm 12, which is denoted by **CDNMG3D**;

(iii) Our proposed 3D inverse consistent model, equipped with the NMG scheme outlined in Algorithm 13, denoted by **ICNMG3D**;

(iv) A state of the art commercial software which we denote by **Eclipse**.

From these results we aim to show how our proposed 3D models perform in terms of accuracy when compared with a commercial software used in hospitals. Moreover, we include a comparison of our proposed models with a standard diffusion model to highlight the fact our proposed models always produce physical transformations even

when the diffusion models fails to do so while maintaining a similar level of accuracy.

We therefore propose to split the comparisons into two parts. First we compare the accuracy of the proposed **CDNMG3D** and **ICNMG3D** models with the results obtained from the **DNMG3D** model. The aim of this comparison is to demonstrate how our proposed models produce similarly accurate results to the **DNMG3D** model with regard to error, while also delivering physically accurate transformations which the **DNMG3D** model does not emphasise. To quantify the accuracy of the registrations we measure four quantities, these are;

(i) Structural similarity (SSIM) [138];

(ii) Relative error defined by

$$Err = \frac{|T_{\boldsymbol{u}} - R|_2^2}{|R|_2^2};$$

(iii) Mean squared error defined by

$$MSE = \frac{1}{N} |T_{\boldsymbol{u}} - R|_2^2$$

where N defines the total number of voxels in the 3D image;

(iv) Minimum determinant of the gradient of the transformation defined by

$$Q_{min} = \det \left(\nabla_3 \boldsymbol{\varphi} \right).$$

From these four measures, we use the $SSIM$, Err and MSE to measure the accuracy of the registration while the Q_{min} value is used to measure the physicality of the transformation. If $Q_{min} > 0$ this implies there is no folding in the transformation (and is therefore physically accurate), while $Q_{min} \leq 0$ implies the transformation contains folding (and is therefore non-physical).

For the second set of comparisons, we compare the **DNMG3D**, **CDNMG3D** and **ICNMG3D** models with the commercial **Eclipse** software. As we mentioned at the beginning of this chapter, validating image registration methods on real lung CT methods is very challenging due to a lack of ground truth data. In order for us to be able to compare the accuracy of the four methods, we look to use the deformation fields to match clinically drawn contours of various features within the lungs. To quantify the accuracy of the contour matching, we use the following three measures;

(i) DICE metric defined by

$$DICE = \frac{2 |A \cap B|}{|A| + |B|}$$

where A, B denote two sets of points;

(ii) Hausdorff distance (HD) defined by

$$HD = \max\{(d(A,B), d(B,A))\}$$

where

$$d(A,B) = \max_{a \in A}\{d(a,B)\}\,,\; d(B,A) = \max_{b \in B}\{d(b,A)\}$$

denote the distance of an element a from set A to any point in set B, and an element b from set B to any point in set A respectively;

(iii) Modified Hausdorff distance (MHD) defined by

$$MHD = \frac{N_a d(A,B) + N_b d(B,A)}{N_a + N_b}$$

where N_a, N_b denote the total number of elements within set A, B respectively and where $d(A,B)$, $d(B,a)$ are the same distances defined in the HD.

Each of these values are computed for contours outlining the following nine features of the lungs;

(i) Body;

(ii) Gross tumour volume (GTV);

(iii) Right lung (RLung);

(iv) Left lung (LLung);

(v) Trachea;

(vi) Carina;

(vii) Oesophagus;

(viii) Heart;

(ix) Spinal column.

For both sets of comparisons we perform all tests on a set of eight 4DCT scans taken from the Hugo database [80], and for each image set we register the image corresponding to peak-exhalation with the image corresponding to peak-inhalation. We also remark in Tables 6.4-6.19. $DICE$ values are within the range $[0,1]$ with 1 indicating perfect overlap and 0 no overlap. In addition, since the HD and MHD values are distance measures, values closer to zero imply better matching of the contours, also where the HD and MHD values are measured in millimetres (mm). Moreover, due to how the **Eclipse** software functions we are unable to obtain results corresponding to the body contours. In addition, we remark the average values shown in Tables 6.4-6.19 are obtained using

all contours except for the body contour in order for comparisons with the **Eclipse** model to be fair.

6.5.1 Accuracy and physicality of the proposed models versus the diffusion model

Here we demonstrate how our two proposed 3D models can achieve a comparable level of accuracy with regard to registration error when compared with a 3D diffusion model, while also maintaining the physical integrity of the transformation, something which the diffusion model is unable to do.

From Table 6.2, we see in the majority of the eight tests our **CDNMG3D** model produces identical results to the **DNMG3D** model, with the results from our **ICNMG3D** model also being similar. For all three models we see for the parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$, the Q_{min} values are positive in all eight tests implying physically accurate results.

When looking at the results in Table 6.3 where we used the parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$ however, we notice the advantage of our proposed models over the **DNMG3D** model, especially our **ICNMG3D** model. In seven of the eight cases, our **ICNMG3D** model achieved the best results in all three error categories ($SSIM/Err/MSE$), although the corresponding values for our **CDNMG3D** model are only marginally worse. However, it is when we look at the Q_{min} columns where the difference between our two proposed models and the **DNMG3D** model becomes most apparent. Here we see in four of the eight tests the **DNMG3D** model failed to produce physically accurate deformations as indicated by the red values in Table 6.3, thus highlighting the problem of models which do not specifically avoid folding. On the other hand, we see both our **CDNMG3D** and **ICNMG3D** models produce physically accurate non-folding deformations in all eight tests.

6.5.2 Comparison of three registration models with the commercial Eclipse model

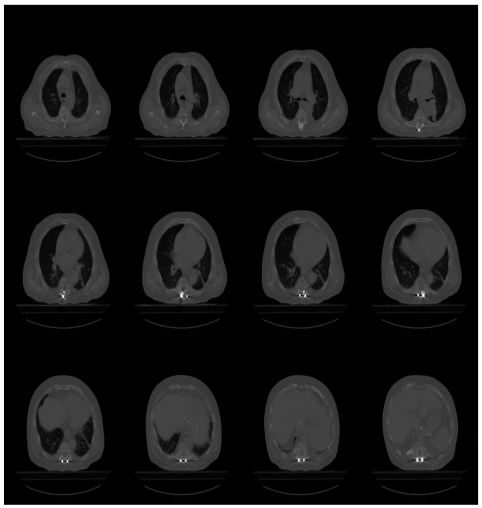
Now we demonstrate how the **DNMG3D**, **CDNMG3D** and **ICNMG3D** models can produce competitive registrations when compare with the state of the are **Eclipse** model. Here we remark values highlighted in bold within Tables 6.4-6.19 indicate instances where our proposed models achieved the best result of all four tested models. When we look at the results shown in Tables 6.4-6.19, we see the **Eclipse** software produces the best results for the majority of cases, however we also see our proposed models and diffusion model produce competitive results in all but a small number of cases. Moreover, when looking at the averages of each measured value, we see our proposed models perform very well in reducing the HD and MHD values when compared with the **Eclipse** model, especially in the examples with parameter $\alpha = \frac{1}{100}$. Although

the **Eclipse** model performs better than our proposed models when comparing the $DICE$ values.

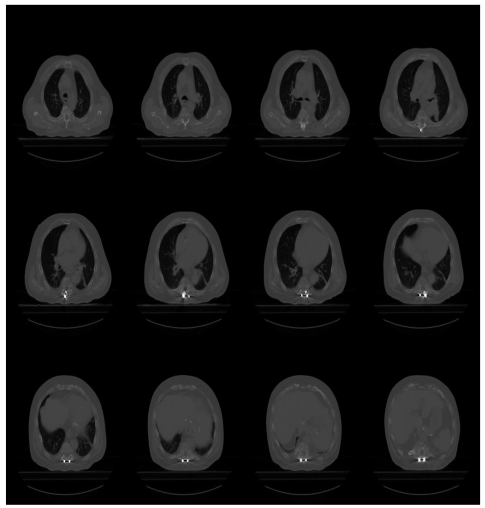
However, we remark the **Eclipse** model uses a pre-affine registration step in order to align the images and obtain an initial guess for the deformable registration step, whereas for our proposed models and diffusion model we use a zero initialisation. Therefore, it would be of interest to develop full multigrid schemes for each of the proposed models to ensure we obtain a good initial guess for the NMG scheme, which consequently will lead to a more accurate registration result.

6.6 Summary

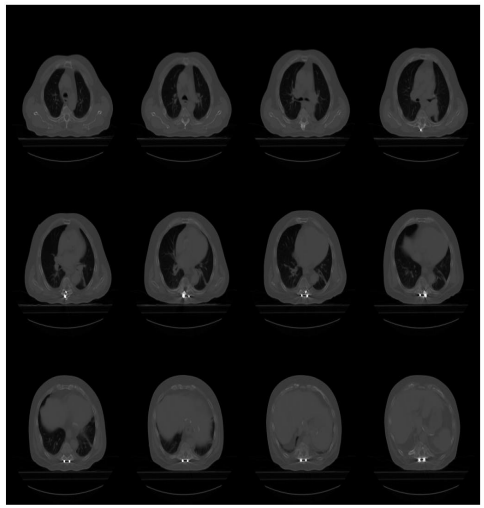
In this chapter we began by formulating the image registration problem in 3D, followed by extending the 2D models discussed in Chapters 4 and 5 into 3D. Next we introduced the FAS-NMG schemes to be implemented for each model along with several potential smoother schemes to be used within the NMG method, before performing a detailed analysis of the key components of the NMG methods. Following this analysis, we showed some preliminary results comparing the three proposed models with a commercial software using eight examples from the Hugo image database [80]. Here we showed how the results from the proposed models where competitive with the commercial Eclipse software, in addition to describing how we could potentially improve our models.



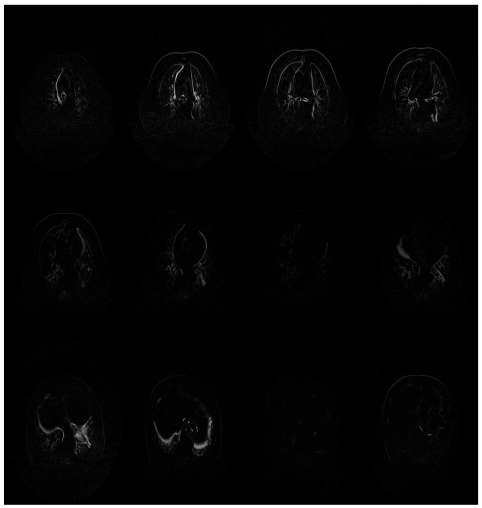
(a) Reference image R.



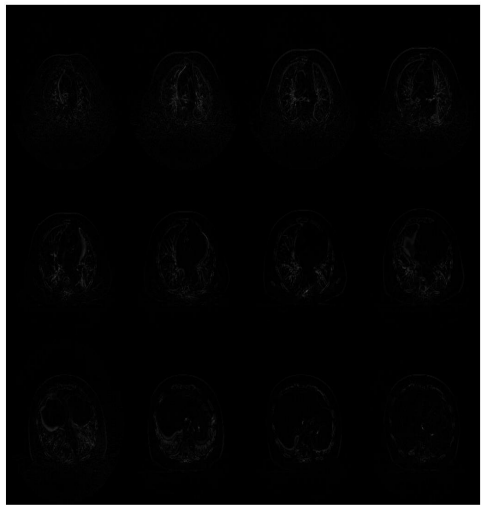
(b) Template image T.



(c) Deformed template image $T_{\boldsymbol{u}}$.



(d) Initial error $|T - R|$.



(e) Final error $|T_{\boldsymbol{u}} - R|$.

Figure 6.2: Patient 1: Registration of (a) and (b) using the standard 3D diffusion model **DNMG3D** with parameter $\alpha = \frac{1}{20}$. Image (c) shows the deformed template image, while images (d) and (e) show the initial and final errors respectively.

	Initial			DNMG3D				CDNMG3D				ICNMG3D			
Image Set	***SSIM***	***Err*** (%)	***MSE***	***SSIM***	***Err*** (%)	***MSE***	Q_{min}	***SSIM***	***Err*** (%)	***MSE***	Q_{min}	***SSIM***	***Err*** (%)	***MSE***	Q_{min}
Patient 1	0.944	3.71	$5.703e-04$	0.958	1.97	$3.030e-04$	0.622	0.958	1.97	$3.030e-05$	0.622	**0.964**	**1.77**	$\mathbf{2.713e-04}$	0.688
Patient 2	0.947	1.86	$2.998e-04$	**0.958**	**1.19**	$\mathbf{1.923e-04}$	0.746	**0.958**	**1.19**	$\mathbf{1.923e-04}$	0.746	0.957	1.25	$2.023e-04$	0.742
Patient 3	0.979	1.98	$1.961e-04$	**0.985**	**1.14**	$\mathbf{1.123e-04}$	0.732	**0.985**	**1.14**	$\mathbf{1.123e-04}$	0.732	**0.985**	1.19	$1.172e-04$	0.719
Patient 4	0.980	0.96	$1.161e-04$	**0.984**	**0.61**	$7.412e-05$	0.828	**0.984**	**0.61**	$7.412e-05$	0.828	**0.984**	**0.61**	$\mathbf{7.353e-05}$	0.834
Patient 5	0.979	0.89	$1.419e-04$	**0.983**	**0.58**	$\mathbf{9.314e-05}$	0.863	**0.983**	**0.58**	$\mathbf{9.314e-05}$	0.863	0.983	0.60	$9.606e-05$	0.848
Patient 6	0.947	2.05	$3.259e-04$	**0.962**	**1.12**	$\mathbf{1.783e-04}$	0.771	**0.962**	**1.12**	$\mathbf{1.783e-04}$	0.771	0.961	1.15	$1.824e-04$	0.746
Patient 7	0.966	2.93	$2.989e-04$	0.973	1.54	$1.576e-04$	0.790	0.973	1.54	$1.576e-04$	0.790	**0.982**	**0.99**	$\mathbf{1.010e-04}$	0.824
Patient 8	0.970	0.96	$1.638e-04$	**0.979**	**0.50**	$\mathbf{8.525e-05}$	0.811	**0.979**	**0.50**	$\mathbf{8.525e-05}$	0.811	**0.979**	**0.50**	$8.618e-05$	0.828

Table 6.2: Example 1: Comparison of forward registrations between three methods on different image sets, in all cases the parameter $\alpha = \frac{1}{20}$ was used and for the inverse consistent model the parameter $\beta = 10^4$ was selected.

	Initial			DNMG3D				CDNMG3D				ICNMG3D			
Image Set	***SSIM***	***Err*** (%)	***MSE***	***SSIM***	***Err*** (%)	***MSE***	Q_{min}	***SSIM***	***Err*** (%)	***MSE***	Q_{min}	***SSIM***	***Err*** (%)	***MSE***	Q_{min}
Patient 1	0.944	3.71	$5.703e-04$	0.966	1.02	$1.567e-04$	−0.704	0.967	1.00	$1.540e-04$	0.040	**0.972**	**0.87**	$\mathbf{1.342e-04}$	0.144
Patient 2	0.947	1.86	$2.998e-04$	0.961	**0.74**	$\mathbf{1.190e-04}$	0.106	0.961	**0.74**	$\mathbf{1.190e-04}$	0.365	**0.962**	**0.74**	$\mathbf{1.194e-04}$	0.121
Patient 3	0.979	1.98	$1.961e-04$	0.988	0.60	$5.892e-05$	−0.088	0.988	0.59	$5.822e-05$	0.408	**0.989**	**0.57**	$\mathbf{5.633e-05}$	0.192
Patient 4	0.980	0.96	$1.161e-04$	**0.987**	**0.39**	$\mathbf{4.471e-05}$	0.348	**0.987**	**0.39**	$4.736e-05$	0.500	0.987	0.40	$4.795e-05$	0.415
Patient 5	0.979	0.89	$1.419e-04$	0.984	0.38	$6.028e-05$	0.271	0.984	0.37	$5.930e-05$	0.519	**0.985**	**0.36**	$\mathbf{5.794e-05}$	0.394
Patient 6	0.947	2.05	$3.259e-04$	0.966	0.62	$9.920e-05$	0.200	0.966	0.63	$9.978e-05$	0.434	**0.967**	**0.60**	$\mathbf{9.617e-05}$	0.057
Patient 7	0.966	2.93	$2.989e-04$	0.980	0.84	$8.578e-05$	−0.049	0.980	0.85	$8.723e-05$	0.415	**0.986**	**0.62**	$\mathbf{6.331e-05}$	0.342
Patient 8	0.970	0.96	$1.638e-04$	0.981	0.34	$5.757e-05$	−0.316	0.981	0.34	$5.757e-05$	0.093	0.983	0.29	$4.919e-05$	0.429

Table 6.3: Example 1: Comparison of forward registrations between three methods on different image sets, in all cases the parameter $\alpha = \frac{1}{100}$ was used and for the inverse consistent model the parameter $\beta = 10^4$ was selected.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.646	10.241	2.223	0.985	6.643	1.854	0.985	6.643	1.854	**0.986**	**5.789**	**1.712**	-	-	-
GTV		0.552	5.177	0.375	0.785	**3.163**	**0.230**	0.785	3.163	0.230	0.781	3.397	0.233	0.889	3.573	0.232
Rlung		0.380	11.119	2.838	0.915	8.795	1.396	0.915	8.795	1.396	0.919	6.001	1.371	0.963	5.445	1.207
Llung		0.371	9.310	2.245	0.912	5.617	**1.285**	0.912	5.617	**1.285**	0.914	**5.140**	1.291	0.966	5.239	1.322
Trachea	$\frac{1}{20}$	0.701	2.222	0.099	0.857	2.144	0.083	0.857	2.144	0.083	0.879	1.881	**0.078**	0.920	1.836	0.086
Carina		0.429	3.894	0.087	0.537	3.005	0.061	0.537	3.005	0.061	0.589	2.480	0.058	0.830	1.808	0.037
Oesophagus		0.405	5.344	0.342	0.430	5.345	0.297	0.430	5.345	0.297	0.438	5.144	0.298	0.545	3.333	0.251
Heart		0.410	6.939	1.228	0.908	5.775	0.737	0.908	5.775	0.737	0.915	4.962	0.756	0.977	4.836	0.693
Spinal Column		0.738	2.773	0.206	0.799	**2.075**	0.180	0.799	**2.075**	0.180	0.786	2.286	**0.179**	0.844	2.935	0.216
Average		0.498	5.847	0.928	0.768	4.490	0.534	0.768	4.490	0.534	0.778	3.911	0.533	0.867	3.251	0.506

Table 6.4: Comparison of the matching of deformed contours between four methods for patient set 1 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.646	10.241	2.223	0.633	8.137	1.331	0.986	8.145	1.308	**0.987**	**5.648**	**1.254**	-	-	-
GTV		0.552	5.177	0.375	0.544	4.736	0.292	0.752	3.333	0.230	0.783	**3.204**	**0.216**	0.889	3.573	0.232
Rlung		0.380	11.119	2.838	0.564	5.481	1.408	0.935	**5.421**	1.255	0.937	5.742	1.242	0.963	5.445	1.207
Llung		0.371	9.310	2.245	0.592	4.936	0.947	0.931	4.997	**0.958**	0.930	**4.722**	1.095	0.966	5.239	1.322
Trachea	$\frac{1}{100}$	0.701	2.222	0.099	0.640	1.432	0.073	0.839	**1.372**	**0.070**	0.861	1.699	0.076	0.920	1.836	0.086
Carina		0.429	3.894	0.087	0.411	2.398	0.058	0.492	2.239	0.055	0.584	2.241	0.056	0.830	1.808	0.037
Oesophagus		0.405	5.344	0.342	0.446	4.817	0.340	0.449	4.772	0.326	0.456	4.829	0.253	0.545	3.333	0.251
Heart		0.4101	6.939	1.228	0.525	5.997	1.091	0.905	4.881	0.870	0.905	4.996	0.745	0.977	4.836	0.693
Spinal Column		0.7381	2.773	0.206	0.591	2.906	0.215	0.813	**2.146**	0.178	0.791	2.451	**0.170**	0.844	2.935	0.216
Average		0.498	5.847	0.928	0.539	4.088	0.533	0.765	3.645	0.493	0.781	3.736	**0.482**	0.867	3.251	0.506

Table 6.5: Comparison of the matching of deformed contours between four methods for patient set 1 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

		Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
Contour	α	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***
Body		0.892	5.574	1.224	**0.989**	**3.557**	0.944	**0.989**	**3.557**	0.944	0.988	3.809	**0.933**	-	-	-
GTV		0.481	4.723	0.526	0.679	2.970	0.303	0.679	2.970	0.303	0.677	2.788	0.341	0.642	2.402	0.272
Rlung		0.544	10.474	2.268	0.935	5.098	1.175	0.935	5.098	1.175	0.923	5.217	1.165	0.970	4.782	1.139
Llung		0.579	8.904	1.807	0.927	4.967	**1.047**	0.927	4.967	**1.047**	0.927	**4.950**	1.069	0.958	5.142	1.128
Trachea	$\frac{1}{20}$	0.395	3.131	0.120	0.826	**1.191**	**0.081**	0.826	**1.191**	**0.081**	0.828	2.017	**0.081**	0.888	2.049	0.085
Carina		0.343	4.306	0.107	0.711	**2.496**	0.060	0.711	**2.496**	0.060	0.706	2.683	0.063	0.741	2.785	0.058
Oesophagus		0.439	6.093	0.568	0.497	5.156	0.359	0.497	5.156	**0.359**	0.500	3.380	0.526	0.528	3.588	0.371
Heart		0.344	8.903	1.239	0.905	4.734	0.798	0.905	4.734	0.798	0.907	5.100	0.831	0.962	4.538	0.737
Spinal Column		0.780	2.806	0.201	0.806	**2.513**	**0.169**	0.806	**2.513**	**0.169**	0.808	2.560	0.175	0.892	3.070	0.198
Average		0.488	6.168	0.855	0.786	3.641	**0.499**	0.786	3.641	**0.499**	0.785	3.587	0.531	0.823	3.545	**0.499**

Table 6.6: Comparison of the matching of deformed contours between four methods for patient set 2 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

		Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
Contour	α	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***	***DICE***	***HD***	***MHD***
Body		0.892	5.574	1.224	0.641	5.641	0.994	**0.989**	3.997	0.994	**0.989**	**3.414**	**0.935**	-	-	-
GTV		0.481	4.723	0.526	0.562	2.553	0.287	**0.718**	2.399	**0.272**	0.698	**2.291**	**0.272**	0.642	2.402	**0.272**
Rlung		0.544	10.474	2.268	0.600	5.885	1.095	0.942	**4.072**	**1.003**	0.937	4.538	1.007	0.970	4.782	1.139
Llung		0.579	8.904	1.807	0.631	5.262	0.964	0.933	**4.496**	1.001	0.934	4.528	**0.990**	0.958	5.142	1.128
Trachea	$\frac{1}{100}$	0.395	3.131	0.120	0.561	1.797	0.079	0.836	**1.442**	**0.071**	0.829	1.464	0.073	0.888	2.049	0.085
Carina		0.343	4.306	0.107	0.623	3.073	0.074	**0.765**	2.639	0.063	0.685	**2.468**	0.061	0.741	2.785	0.058
Oesophagus		0.439	6.093	0.568	0.499	5.691	0.454	0.499	5.691	0.454	0.505	4.771	0.462	0.528	3.588	0.371
Heart		0.344	8.903	1.239	0.546	6.180	0.876	0.918	4.587	0.753	0.908	4.701	0.793	0.962	4.538	0.737
Spinal Column		0.780	2.806	0.201	0.596	2.614	0.201	0.802	**2.537**	0.175	0.809	2.567	**0.169**	0.892	3.070	0.198
Average		0.488	6.168	0.855	0.577	4.132	0.504	0.801	3.483	**0.474**	0.788	**3.416**	0.478	0.823	3.545	0.499

Table 6.7: Comparison of the matching of deformed contours between four methods for patient set 2 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.903	8.359	1.041	**0.982**	**5.198**	0.937	**0.982**	**5.198**	0.937	**0.982**	5.377	**0.713**	-	-	-
GTV		0.382	4.109	0.211	0.747	2.198	0.118	**0.747**	2.198	0.118	0.744	2.690	0.123	0.713	1.822	0.098
Rlung		0.595	9.637	1.862	0.933	4.798	0.969	0.933	4.798	0.969	0.929	**4.748**	**0.975**	0.966	5.413	1.155
Llung		0.587	8.459	1.795	0.912	**4.921**	**1.044**	0.912	**4.921**	**1.044**	0.909	4.923	1.067	0.952	5.418	1.204
Trachea	$\frac{1}{20}$	0.191	5.235	0.244	0.232	2.450	0.104	0.232	2.450	0.104	0.232	2.485	0.104	0.420	2.203	0.100
Carina		0.179	4.577	0.197	0.201	2.332	0.068	0.201	2.332	0.068	0.198	2.342	0.068	0.799	1.755	0.033
Oesophagus		0.545	4.252	0.234	0.501	**3.474**	**0.198**	0.501	**3.474**	**0.198**	0.503	3.686	0.201	0.511	4.931	0.485
Heart		0.489	6.313	0.701	0.952	3.643	**0.410**	0.952	3.643	**0.410**	0.952	**3.633**	0.448	0.965	4.015	0.524
Spinal Column		0.747	2.944	0.168	0.735	**2.519**	**0.139**	0.735	**2.519**	**0.139**	0.732	2.645	0.143	0.847	3.378	0.192
Average		0.464	5.691	0.677	0.652	**3.292**	**0.381**	0.652	**3.292**	**0.381**	0.650	3.394	0.772	3.617	0.474	

Table 6.8: Comparison of the matching of deformed contours between four methods for patient set 3 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.903	8.359	1.041	0.633	5.276	0.993	**0.984**	5.706	0.709	0.983	**3.718**	**0.676**	-	-	-
GTV		0.382	4.109	0.211	0.670	2.177	0.084	**0.822**	**1.499**	**0.084**	0.754	1.703	0.097	0.713	1.822	0.098
Rlung		0.595	9.637	1.862	0.596	5.374	0.994	0.949	4.582	0.904	0.947	**4.546**	**0.887**	0.966	5.413	1.155
Llung		0.587	8.459	1.795	0.611	5.248	1.046	0.932	**4.905**	**0.987**	0.926	4.992	0.992	0.952	5.418	1.204
Trachea	$\frac{1}{100}$	0.191	5.235	0.244	0.210	5.157	0.241	0.210	2.370	**0.098**	0.232	2.467	0.100	0.420	2.203	0.100
Carina		0.181	4.577	0.197	0.173	4.538	0.199	0.191	2.287	0.072	0.190	2.303	0.070	0.799	1.755	0.033
Oesophagus		0.545	4.252	0.234	0.551	3.418	0.188	**0.552**	**3.208**	**0.188**	0.536	3.261	**0.188**	0.511	4.931	0.485
Heart		0.489	6.313	0.701	0.569	4.052	0.421	0.928	3.869	0.385	0.932	3.874	0.414	0.965	4.015	0.524
Spinal Column		0.747	2.944	0.168	0.544	2.444	0.163	0.724	**2.298**	**0.143**	0.729	2.498	0.148	0.847	3.378	0.192
Average		0.464	5.691	0.677	0.491	4.025	0.417	0.664	**3.127**	**0.358**	0.565	3.206	0.362	0.772	3.617	0.474

Table 6.9: Comparison of the matching of deformed contours between four methods for patient set 3 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.906	8.782	1.311	**0.984**	**6.647**	1.151	**0.984**	**6.647**	1.151	0.984	6.618	**0.924**	-	-	-
GTV		0.274	3.358	0.139	**0.696**	2.015	**0.100**	**0.696**	2.015	**0.100**	0.690	**1.996**	**0.100**	0.244	2.202	0.104
Rlung		0.721	6.427	1.243	0.952	**5.464**	**0.991**	0.952	**5.464**	**0.991**	0.949	5.479	0.993	0.970	5.601	1.214
Llung		0.975	6.322	0.140	0.953	2.426	0.399	0.953	2.426	0.399	0.952	**2.317**	**0.397**	0.972	5.121	1.371
Trachea	$\frac{1}{20}$	0.848	0.987	0.043	0.821	1.137	0.039	0.821	1.137	0.039	0.829	**1.129**	**0.038**	0.902	1.720	0.046
Carina		0.521	3.440	0.072	0.756	2.584	0.052	0.756	2.584	0.052	0.730	2.628	0.053	0.841	1.855	0.038
Oesophagus		0.624	4.212	0.294	0.535	3.956	0.282	0.535	3.956	0.282	0.544	0.971	0.282	0.589	3.375	0.209
Heart		0.622	5.855	0.788	0.910	5.534	0.694	0.910	5.534	0.694	0.906	5.539	0.695	0.956	4.935	0.655
Spinal Column		0.823	2.461	0.128	0.808	2.047	**0.118**	0.808	2.047	**0.118**	0.811	**2.042**	0.119	0.919	2.637	0.146
Average		0.676	4.133	0.356	**0.804**	3.145	**0.334**	**0.804**	3.145	**0.334**	0.801	**3.134**	0.335	0.799	3.431	0.473

Table 6.10: Comparison of the matching of deformed contours between four methods for patient set 4 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.906	8.782	1.311	0.658	11.052	1.052	**0.986**	**5.318**	**1.052**	0.985	5.428	1.055	-	-	-
GTV		0.274	3.358	0.139	0.477	3.266	0.130	**0.698**	**1.924**	**0.101**	0.695	1.935	0.101	0.244	2.202	0.104
Rlung		0.721	6.427	1.243	0.627	5.274	0.822	0.955	**5.178**	**0.887**	0.951	5.198	0.892	0.970	5.601	1.214
Llung		0.975	1.322	0.140	0.677	4.193	0.591	0.958	**2.590**	0.588	0.955	2.867	**0.577**	0.972	5.121	1.371
Trachea	$\frac{1}{100}$	0.848	0.987	0.043	0.667	0.983	0.033	0.823	**1.018**	**0.037**	0.819	1.054	**0.037**	0.902	1.720	0.046
Carina		0.521	3.440	0.072	0.653	1.837	0.035	0.745	1.935	0.040	0.732	2.020	0.39	0.841	1.855	0.038
Oesophagus		0.624	4.212	0.294	0.543	3.910	0.277	0.543	3.910	0.277	0.545	0.900	0.276	0.589	3.375	0.209
Heart		0.622	5.855	0.788	0.567	4.295	0.586	0.910	4.295	0.584	0.910	5.178	0.587	0.956	4.935	0.655
Spinal Column		0.823	2.461	0.128	0.596	2.251	0.122	0.812	1.977	0.116	0.811	1.977	0.117	0.919	2.637	0.146
Average		0.676	4.133	0.356	0.601	3.251	0.325	**0.806**	**2.853**	**0.329**	0.802	3.016	**0.329**	0.799	3.431	0.473

Table 6.11: Comparison of the matching of deformed contours between four methods for patient set 4 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.863	8.155	1.743	**0.987**	5.420	1.252	**0.987**	5.420	1.252	0.986	**5.366**	**1.230**	-	-	-
GTV		0.449	3.502	0.401	0.860	**2.694**	**0.298**	0.860	**2.694**	**0.298**	**0.861**	2.757	0.305	0.831	2.942	0.314
Rlung		0.513	8.988	2.329	0.923	5.561	1.535	0.923	5.561	1.535	0.918	5.675	**0.549**	0.963	5.236	1.492
Llung		0.635	7.257	1.825	0.935	**5.458**	**1.441**	0.935	**5.458**	**1.441**	0.933	5.540	1.452	0.963	5.575	1.433
Trachea	$\frac{1}{20}$	0.733	1.732	0.091	0.788	**1.324**	0.088	0.788	**1.324**	0.088	0.823	1.670	0.089	0.916	1.595	0.084
Carina		0.573	3.201	0.073	0.653	2.391	0.051	0.653	2.391	0.051	0.664	2.532	0.052	0.740	2.289	0.048
Oesophagus		0.599	3.397	0.310	0.548	3.151	0.295	0.548	3.151	0.295	0.545	3.167	0.295	0.680	2.571	0.221
Heart		0.471	6.694	1.223	0.899	4.725	**0.876**	0.899	4.725	**0.876**	0.888	4.905	0.890	0.911	1.181	**0.876**
Spinal Column		0.784	2.862	0.183	0.847	**2.002**	**0.173**	0.847	**2.002**	**0.173**	0.836	2.706	0.175	0.921	2.608	0.180
Average		0.595	4.704	0.804	0.807	3.413	0.595	0.807	3.413	0.595	0.809	3.619	**0.476**	0.866	3.001	0.581

Table 6.12: Comparison of the matching of deformed contours between four methods for patient set 5 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.863	8.155	1.743	0.621	5.638	1.451	**0.987**	5.368	1.076	**0.987**	**5.357**	**1.055**	-	-	-
GTV		0.449	3.502	0.401	0.614	3.398	0.320	0.864	2.680	**0.274**	**0.865**	**2.653**	0.286	0.831	2.942	0.314
Rlung		0.513	8.988	2.329	0.569	6.100	1.496	0.939	4.978	**1.344**	0.932	**4.826**	1.364	0.963	5.236	1.492
Llung		0.635	7.257	1.825	0.625	4.503	1.082	0.948	**4.196**	**1.244**	0.940	4.607	1.293	0.963	5.575	1.433
Trachea	$\frac{1}{100}$	0.733	1.732	0.091	0.566	0.993	0.065	0.799	**1.263**	**0.075**	0.798	1.491	0.081	0.916	1.595	0.084
Carina		0.573	3.201	0.073	0.575	1.991	0.048	0.661	2.240	0.056	0.676	**1.934**	**0.047**	0.740	2.289	0.048
Oesophagus		0.599	3.397	0.310	0.524	3.187	0.295	0.524	3.187	0.295	0.568	3.213	0.289	0.680	2.571	0.221
Heart		0.471	6.694	1.223	0.552	5.813	1.059	0.895	4.221	**0.805**	0.899	4.377	0.833	0.911	1.181	0.876
Spinal Column		0.784	2.862	0.183	0.625	2.399	0.184	0.822	**2.182**	0.173	0.830	2.531	**0.172**	0.921	2.608	0.180
Average		0.595	4.704	0.804	0.581	3.548	0.569	0.807	3.118	**0.533**	0.814	3.092	0.546	0.866	3.001	0.581

Table 6.13: Comparison of the matching of deformed contours between four methods for patient set 5 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.844	5.951	1.381	0.984	3.619	1.146	0.984	3.619	1.146	**0.985**	**3.570**	**0.966**	-	-	-
GTV		0.327	8.433	1.162	0.783	4.770	0.641	0.783	4.770	0.641	0.773	4.853	0.648	0.898	4.342	0.627
Rlung		0.453	8.383	2.457	0.917	5.156	1.299	0.917	5.156	1.299	0.914	5.223	1.299	0.962	5.038	1.268
Llung		0.439	8.199	2.501	0.936	5.026	1.506	0.936	5.026	1.506	0.937	5.083	1.536	0.965	4.879	1.408
Trachea	$\frac{1}{20}$	0.386	3.511	0.238	0.831	**2.089**	**0.145**	0.831	**2.089**	**0.145**	**0.836**	2.186	0.148	0.5808	2.199	0.158
Carina		0.139	5.101	0.137	0.436	2.966	0.061	0.436	2.966	0.061	0.398	2.941	0.061	0.696	2.439	0.052
Oesophagus		0.381	5.896	0.474	0.535	4.503	0.384	0.535	4.503	0.384	0.538	4.819	0.390	0.627	3.634	0.265
Heart		0.192	8.568	1.771	0.848	6.149	0.969	0.848	6.149	0.969	0.849	6.178	0.979	0.948	4.990	0.874
Spinal Column		0.655	4.472	0.287	0.821	3.418	**0.240**	0.821	3.418	**0.240**	0.818	**3.281**	0.248	0.839	3.815	0.280
Average		0.372	6.520	1.128	0.763	4.260	0.657	0.763	4.260	0.657	0.758	4.321	0.664	0.814	3.917	0.617

Table 6.14: Comparison of the matching of deformed contours between four methods for patient set 6 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.844	5.951	1.381	0.642	3.818	1.338	**0.989**	3.818	1.006	0.985	**3.146**	**0.901**	-	-	-
GTV		0.327	8.433	1.162	0.563	4.400	0.591	0.828	**3.917**	**0.575**	0.815	4.186	0.625	0.898	4.342	0.627
Rlung		0.453	8.383	2.457	0.609	3.781	0.973	0.952	**3.691**	**1.058**	0.939	4.299	1.118	0.962	5.038	1.268
Llung		0.439	8.199	2.501	0.642	5.345	1.075	0.947	4.379	1.245	0.945	**4.199**	**1.188**	0.965	4.879	1.408
Trachea	$\frac{1}{100}$	0.386	3.511	0.238	0.588	2.101	0.143	0.754	2.126	**0.139**	**0.857**	**2.030**	**0.139**	0.581	2.199	0.158
Carina		0.139	5.101	0.137	0.584	2.286	0.068	0.666	**1.907**	0.057	0.650	2.210	0.058	0.696	2.439	0.052
Oesophagus		0.381	5.896	0.474	0.542	3.839	0.301	0.542	**3.052**	**0.242**	0.573	3.428	0.316	0.627	3.634	0.265
Heart		0.192	8.568	1.771	0.594	6.636	1.023	0.906	5.229	0.826	0.870	6.038	0.919	0.948	4.990	0.874
Spinal Column		0.655	4.472	0.287	0.584	4.282	0.275	0.795	3.547	0.246	0.822	**3.421**	**0.242**	0.839	3.815	0.280
Average		0.372	6.520	1.128	0.588	4.084	0.556	0.799	**3.481**	**0.549**	0.809	3.726	0.576	0.814	3.917	0.617

Table 6.15: Comparison of the matching of deformed contours between four methods for patient set 6 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.778	9.113	1.670	0.975	8.386	1.496	0.975	8.386	1.496	**0.976**	**8.049**	**1.453**	-	-	-
GTV		0.772	1.899	0.148	0.720	2.441	0.155	0.720	2.441	0.155	0.737	**2.370**	**0.138**	0.777	2.692	0.146
Rlung		0.448	9.491	1.812	0.933	5.424	1.152	0.933	5.424	1.152	0.941	5.403	**1.122**	0.975	5.304	1.139
Llung		0.436	8.489	1.963	0.932	5.814	1.172	0.932	5.814	1.172	0.940	5.659	1.140	0.970	5.207	1.138
Trachea	$\frac{1}{20}$	0.669	2.049	0.078	0.794	1.603	0.071	0.794	1.603	0.071	0.803	**1.321**	0.073	0.907	1.776	0.070
Carina		0.404	3.047	0.053	0.433	2.320	0.039	0.433	2.320	0.039	0.574	2.211	0.037	0.683	1.715	0.034
Oesophagus		0.714	3.278	0.176	0.653	2.572	0.130	0.653	2.572	0.130	0.561	**2.504**	**0.129**	0.599	3.312	0.234
Heart		0.537	6.647	0.867	0.887	5.873	0.680	0.887	5.873	0.680	0.896	5.195	0.671	0.960	4.746	0.633
Spinal Column		1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000
Average		0.623	4.363	0.410	0.794	3.256	0.425	0.794	3.256	0.425	0.807	**3.083**	**0.414**	0.859	3.094	0.496

Table 6.16: Comparison of the matching of deformed contours between four methods for patient set 7 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.778	9.113	1.670	0.642	8.508	1.514	**0.975**	**7.558**	**1.220**	**0.975**	8.049	1.356	-	-	-
GTV		0.772	1.899	0.148	0.607	2.178	0.162	0.722	2.523	0.162	0.732	**2.357**	**0.144**	0.777	2.692	0.146
Rlung		0.448	9.491	1.812	0.607	4.258	0.905	0.937	4.677	0.977	0.938	**4.580**	**0.971**	0.975	5.304	1.139
Llung		0.436	8.489	1.963	0.588	5.713	1.033	0.938	4.720	0.994	0.939	**4.636**	**0.987**	0.970	5.207	1.138
Trachea	$\frac{1}{100}$	0.669	2.049	0.078	0.614	1.480	0.066	0.786	**1.771**	0.078	0.791	1.876	0.076	0.907	1.776	0.070
Carina		0.404	3.047	0.053	0.459	1.547	0.030	0.529	1.383	**0.027**	0.714	**1.339**	0.028	0.683	1.715	0.034
Oesophagus		0.714	3.278	0.176	0.505	3.796	0.204	**0.626**	**2.827**	**0.170**	0.546	3.250	0.181	0.599	3.312	0.234
Heart		0.537	6.647	0.867	0.592	4.970	0.555	0.910	**1.292**	**0.559**	0.912	4.171	0.561	0.960	4.746	0.633
Spinal Column		1.000	0.000	0.000	0.797	1.748	0.114	1.000	0.000	0.000	1.000	0.000	0.000	1.000	0.000	0.000
Average		0.623	4.363	0.416	0.596	3.211	0.384	0.806	**2.399**	0.371	0.822	2.776	**0.639**	0.859	3.094	0.496

Table 6.17: Comparison of the matching of deformed contours between four methods for patient set 7 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.646	10.241	2.223	**0.990**	6.053	**1.909**	**0.990**	6.053	**1.909**	**0.990**	**6.001**	1.927	-	-	-
GTV		0.552	5.177	0.375	0.791	**3.102**	**0.232**	0.791	**3.102**	**0.232**	0.785	3.170	0.235	0.868	3.884	0.517
Rlung		0.380	11.119	2.838	0.924	6.020	1.420	0.924	6.020	1.420	0.923	6.059	1.433	0.959	4.995	1.102
Llung		0.371	9.310	2.245	0.924	5.536	1.352	0.924	5.536	1.352	0.920	5.541	1.379	0.949	5.400	1.223
Trachea	$\frac{1}{20}$	0.701	2.222	0.099	0.878	1.877	**0.089**	0.878	1.877	**0.089**	0.880	1.888	0.090	0.869	1.538	0.096
Carina		0.429	3.894	0.087	0.587	2.223	0.052	0.587	2.223	0.052	0.585	2.342	0.054	0.807	1.761	0.044
Oesophagus		0.405	5.344	0.342	0.457	**2.589**	0.291	0.457	2.589	0.291	0.453	2.734	0.308	0.602	3.904	0.280
Heart		0.410	6.939	1.228	**0.950**	**4.848**	**0.785**	**0.950**	**4.848**	**0.785**	0.948	5.100	0.829	0.933	5.643	0.928
Spinal Column		0.738	2.773	0.206	0.833	**2.094**	**0.175**	0.833	**2.094**	**0.175**	0.834	2.264	0.180	0.889	3.594	0.193
Average		0.498	5.847	0.928	0.793	**3.563**	0.550	0.793	**3.563**	0.550	0.791	3.637	0.564	0.860	3.840	0.548

Table 6.18: Comparison of the matching of deformed contours between four methods for patient set 8 with parameters $\alpha = \frac{1}{20}$, $\beta = 10^4$.

Contour	α	Initial			DNMG3D			CDNMG3D			ICNMG3D			Eclipse		
		DICE	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*	*DICE*	*HD*	*MHD*
Body		0.646	10.241	2.223	0.583	9.285	2.101	0.989	7.621	1.929	**0.990**	**6.238**	**1.906**	-	-	-
GTV		0.552	5.177	0.375	0.578	4.267	0.285	0.782	3.520	**0.235**	0.792	**3.365**	0.242	0.868	3.884	0.517
Rlung		0.380	11.119	2.838	0.408	8.641	2.341	0.933	5.956	1.456	0.925	6.050	1.425	0.959	4.995	1.102
Llung		0.371	9.310	2.245	0.513	7.644	1.717	0.942	5.446	1.323	0.920	5.494	1.346	0.949	5.400	1.223
Trachea	$\frac{1}{20}$	0.701	2.222	0.099	0.634	2.001	0.122	0.846	1.777	0.122	0.891	1.736	0.090	0.869	1.538	0.096
Carina		0.429	3.894	0.087	0.433	3.208	0.073	0.513	2.216	0.053	0.577	2.149	0.050	0.807	1.761	0.044
Oesophagus		0.405	5.344	0.342	0.447	4.327	0.266	0.447	4.327	0.266	0.456	2.363	0.268	0.602	3.904	0.280
Heart		0.410	6.939	1.228	0.552	5.356	0.869	**0.951**	4.951	0.869	**0.951**	**4.814**	**0.762**	0.933	5.643	0.928
Spinal Column		0.738	2.773	0.206	0.606	2.933	0.224	0.833	2.244	0.182	0.836	**2.139**	**0.175**	0.889	3.594	0.193
Average		0.498	5.847	0.928	0.523	4.797	0.737	0.781	3.805	0.563	0.794	**3.514**	**0.545**	0.860	3.840	0.548

Table 6.19: Comparison of the matching of deformed contours between four methods for patient set 8 with parameters $\alpha = \frac{1}{100}$, $\beta = 10^4$.

Chapter 7

Conclusions & future research

This thesis has demonstrated the author's work of three image registration models, each with their own fast non-linear multigrid solver, for use in the registration of lung CT images. Of the three proposed models, two of these models addressed the problem of physically accurate registrations by ensuring the resulting deformations where diffeomorphic. Each of the discussed models where then extended to register sets of 3D images, in addition to demonstrating how they can be used in applications within oncology.

7.1 Conclusion

First in Chapter 4, we proposed an improved non-linear multigrid method over the one originally proposed by Chumchob and Chen in [33]. This was achieved by performing a more accurate analysis of the Chumchob-Chen multigrid scheme, along with proposing an alternative solver for use on the coarsest grid level. In addition, we proposed an extension to the Chumchob-Chen model to prevent any folding within the deformation, thus ensuring we obtained image registration results which were physically accurate, by incorporating an additional constraint into the model. We then further extended this new constrained model to improve the accuracy of the registrations, even in the case of severe folding, along with robustness to the choice of weighting parameter. Through experimental results, we demonstrated how our modifications to the Chumchob-Chen model, resulted in vast improvements to the convergence of the multigrid scheme, accuracy of registration and CPU time in addition to the physicality of the deformations.

Second in Chapter 5, we considered an alternate method of achieving diffeomorphic image registration results by looking at inverse consistent registration models. In particular, we focused on the inverse consistent model proposed by Christensen and Johnson in [28], and proposed a solution to the problem of CPU cost resulting from the direct inversion of the non-linear inverse consistency constraint. Our solution was to

approximate this constraint using a linear expression, we then further reduced the CPU cost by implementing a fast non-linear multigrid scheme with three potential smoother schemes. Next we demonstrated the effectiveness of the proposed inverse consistent model using three real lung CT images and a set of synthetic hand X-ray images. From these tests we showed how our proposed model can achieve accurate registrations, with regard to both error and physical transformations, in addition to demonstrating how the multigrid scheme resulted in fast CPU times.

Finally in Chapter 6, we extended the three proposed models from Chapters 4 and 5, along with the corresponding non-linear multigrid schemes, into 3D. Using these 3D models we presented some preliminary results comparing the proposed models to a state of the art commercial software, which is currently used in hospitals. To do this we took eight sets of 3D images from the Hugo database [80], and used three different metrics to measure the accuracy of matching contours deformed from the template image, using the deformation obtained from the registration, to the reference image. For these tests we used contours of nine different features from the scans, which were drawn by a radiologist. From these results we say the proposed models showed promise for use in oncology applications by producing results comparable to the commercial software.

7.2 Future research

The work which has been presented in this thesis has several different directions we can pursue in the future. We now discuss some of these potential avenues of research:

Multi-modal image registration. As we mentioned in the beginning of Chapter 6, it is common practice in oncology to use information from scans taken using different imaging modalities for patient treatment. It would therefore be of interest to develop multi-modal registration models which incorporate the diffeomorphic properties of the models proposed in this thesis, in addition to the proposed fast non-linear multigrid methods.

High order image registration models. The commercial Eclipse software we discussed in Chapter 6 requires an affine registration step to align the images, before the deformable registration can be performed. Moreover, the three proposed models discussed within this thesis all used a diffusion regularisation term, which is first order and therefore penalises affine transformations. Thus, it would be of interest to replace the first order diffusion regulariser with a second order regulariser, such as the linear curvature regulariser, in order to avoid requiring a pre-affine registration step.

Full multigrid method. The multigrid methods which we described for each of the proposed models, were all based upon the V-cycle multigrid scheme which starts on the full discrete image with zero initial guess before coarsening. It would therefore be

beneficial to implement a full multigrid method for the proposed models, which works from the coarse grid to provide a very good initial guess before starting the V-cycle scheme. As a result such a method would lead to a more accurate registration model.

Appendix A

Optimised version of Algorithm 8

In our constrained NMG, we check to see whether the constraint in (4.41) has been satisfied after the final post-smoothing step and solver step. While checking the constraint after the coarsest solver step is inexpensive computationally owing to the very small grid size, this is not the case when checking after the post-smoothing step. For each interior point Algorithm 8 needs to solve eight inverse problems which, even though we are only using 3×3 matrices, become very expensive on larger grids thus leading to a significant increase in CPU time. We now look to exploit the structure and commonality between different interior points, of the matrices A_l, to create an optimised version of Algorithm 8. First we look at the relation of the matrices A_l at the first interior point $(2,2)$ and a general interior point (i,j). Looking at the matrix A_1, we see

$$\textbf{At } (2,2).\ A_1 = \begin{bmatrix} 1 & h & h \\ 1 & 2h & h \\ 1 & h & 2h \end{bmatrix}, \ \textbf{At } (i,j).\ \tilde{A}_1 = \begin{bmatrix} 1 & (i-1)h & (j-1)h \\ 1 & ih & (j-1)h \\ 1 & (i-1)h & jh \end{bmatrix}$$

since

$$\big((x_1)_2, (x_2)_2\big) = (h,h),\ \big((x_1)_i, (x_2)_j\big) = \big((i-1)h, (j-1)h\big).$$

Then $\tilde{A}_1$ can be written in the following way

$$\begin{aligned} \tilde{A}_1 &= \begin{bmatrix} 1 & (x_1)_2 + (i-2)h & (x_2)_2 + (j-2)h \\ 1 & (x_1)_3 + (i-2)h & (x_2)_2 + (j-2)h \\ 1 & (x_1)_2 + (i-2)h & (x_2)_3 + (j-2)h \end{bmatrix} \\ &= A_1 + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \big[0, (i-2)h, (j-2)h\big] \\ &= A_1 + \boldsymbol{p}\boldsymbol{q}^T \end{aligned} \tag{A.1}$$

with

$$\boldsymbol{p} = \big[1,1,1\big]^T,\ \boldsymbol{q} = \big[0,(i-2)h,(j-2)h\big]^T.$$

The matrices $\tilde{A}_l$ for the remaining triangles can be written in similar ways to (A.1), thus we can write

$$\tilde{A}_l = A_l + \boldsymbol{p}\boldsymbol{q}^T$$

with $\boldsymbol{p}$, $\boldsymbol{q}$ as before. Therefore the inverse

$$\tilde{A}_l^{-1} = \big[A_l + \boldsymbol{p}\boldsymbol{q}^T\big]^{-1}$$

Algorithm 14 $Q_{min} = OptFEM(\boldsymbol{u}^h, n, h)$

for $l = 1, \ldots, 4$ **do**
  Compute matrices A_l corresponding to first interior point $(2, 2)$
  Compute inverse matrices A_l^{-1}
  Compute second and third components of $A_l^{-1}\boldsymbol{p} \rightarrow \omega_{p\,l}(2)$, $\omega_{p\,l}(3)$
end for
for $i = 2, \ldots, n-1$ **do**
  for $j = 2, \ldots, n-1$ **do**
  Compute second and third components of $\boldsymbol{q}^T \rightarrow q_2 = (i-1)h$, $q_3 = (j-1)h$
    for $l = 1, \ldots, 4$ **do**
  Compute μ_l
  Compute second and third components of $\omega_{u_1\,l}$, $\omega_{u_2\,l} \rightarrow \omega_{u_1\,l}(2)$, $\omega_{u_1\,l}(3)$, $\omega_{u_2\,l}(2)$, $\omega_{u_2\,l}(3)$
  Determine coefficients $s_{l\,u_1}$, $t_{l\,u_1}$, $s_{l\,u_2}$, $t_{l\,u_2}$ using (A.3)
  Compute determinant for triangle $T_l \rightarrow \tilde{Q}_l = (1 + s_{l\,u_1})(1 + t_{l\,u_2}) - t_{l\,u_1} s_{l\,u_2}$
    end for
  Assign minimum $\tilde{Q}$ to be entry $(Q_{ij}) \rightarrow (Q_{ij}) = \min\{\tilde{Q}_1, \ldots, \tilde{Q}_4\}$
  end for
end for
  Take minimum entry in $\boldsymbol{Q}$ to be minimum determinant value $\rightarrow Q_{min} = \min\{\boldsymbol{Q}\}$

at a general discrete interior point, can be computed using the Sherman-Morrison formula [5] given by the following theorem:

Theorem A.0.1. *(Sherman-Morrison) Suppose $A \in \mathbb{R}^{n\times n}$ is an invertible matrix, and $\boldsymbol{p}$, $\boldsymbol{q} \in \mathbb{R}^{n\times 1}$ are column vectors. Then $\left[A + \boldsymbol{p}\boldsymbol{q}^T\right]$ is invertible*

$$\iff 1 + \boldsymbol{q}^T A^{-1}\boldsymbol{p} \neq 0.$$

If $\left[A + \boldsymbol{p}\boldsymbol{q}^T\right]$ is invertible, then its inverse is given by

$$\left[A + \boldsymbol{p}\boldsymbol{q}^T\right]^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{p}\boldsymbol{q}^T A^{-1}}{1 + \boldsymbol{q}^T A^{-1}\boldsymbol{p}} \tag{A.2}$$

where $\boldsymbol{p}\boldsymbol{q}^T$ denotes the outer product of the vectors $\boldsymbol{p}$, $\boldsymbol{q}$.

It can be shown

$$\boldsymbol{q}^T A_l^{-1}\boldsymbol{p} = 0 \;\forall\; l = 1, \ldots, 4$$

therefore the invertibility condition from Theorem A.0.1 holds for every interior (i, j) for $i, j = 2, \ldots, n-1$ and thus the matrices $\left[A_l + \boldsymbol{p}\boldsymbol{q}^T\right]^{-1}$ are invertible for each $l = 1, \ldots, 4$. Then we can use Theorem A.0.1 to re-write the inverses $\tilde{A}_l^{-1}$ as

$$\tilde{A}_l^{-1} = \left[A_l + \boldsymbol{p}\boldsymbol{q}^T\right]^{-1} = A_l^{-1} - \frac{A_l^{-1}\boldsymbol{p}\boldsymbol{q}^T A_l^{-1}}{1 + \boldsymbol{q}^T A_l^{-1}\boldsymbol{p}}.$$

Next we use the fact we need only determine the $s_{l\,u_m}$, $t_{l\,u_m}$ coefficients where $m = 1, 2$, and so our original inverse problem (4.44) reduces to the following scalar equations

$$\begin{aligned} s_{l\,u_1} &= \omega_{u_1\,l}(2) - \mu_l \omega_{u_1\,l}(2),\; t_{l\,u_1} = \omega_{u_1\,l}(3) - \mu_l \omega_{u_1\,l}(3), \\ s_{l\,u_2} &= \omega_{u_2\,l}(2) - \mu_l \omega_{u_2\,l}(2),\; t_{l\,u_2} = \omega_{u_2\,l}(3) - \mu_l \omega_{u_2\,l}(3), \end{aligned} \tag{A.3}$$

Image Size (n^2)	**Unoptimised Time (s)**	**Optimised Time (s)**
256^2	4.46	0.17
512^2	17.87	0.61
1024^2	71.53	2.40
2048^2	306.23	9.90

Table A.1: Table showing the comparison of CPU times per iteration between old unoptimised FEM code and new optimised FEM code.

where

$$\mu_l = \frac{(\omega_{pl}(2)q_2 + \omega_{pl}(3)q_3)}{1 + (\omega_{pl}(2)q_2 + \omega_{pl}(3)q_3)}$$

and $\omega_{pl}(2)$, $\omega_{pl}(3)$, q_2, q_3, $\omega_{u_m l}(2)$, $\omega_{u_m l}(3)$ denote the second and third components of

$$\omega_{pl} = A_l^{-1}\boldsymbol{p}\boldsymbol{q}^T,\ \omega_{u_m l} = A_l^{-1}\boldsymbol{v}_{ml}$$

respectively.

Finally we show in Table A.1 how much speed up can be achieved for a simple example. Clearly Algorithm 14 uses up to 30 times less CPU when compared with Algorithm 8. Therefore the key message is that per checking step across the entire grid only simple matrix-vector products are needed, if we invert matrices A_l^{-1} at the first pixel and then re-use them. Hence our optimised version of Algorithm 8 can be expressed by Algorithm 14.

Bibliography

[1] M.A. Admiraal, D. Schuring, and C.W. Hurkmans. Dose calculations accounting for breathing motion in stereotactic lung radiotherapy based on 4d-ct and the internal target volume. *Radiotherapy and Oncology*, 86(1):55–60, 2008.

[2] G. Auzias, O. Colliot, J.A Glaunès, M. Perrot, J.F. Mangin, A. Trouvé, and S. Baillet. Diffeomorphic brain registration under exhaustive sulcal constraints. *IEEE Transactions on Medical Imaging*, 30(6):1214–1227, 2011.

[3] I. Babuška, U. Banerjee, and J.E. Osborn. Generalized finite element methods - main ideas, results and perspective. *International Journal of Computational Methods*, 1(1):67–103, 1999.

[4] R. Bajscy and S. Kovačič. Multiresolution elastic matching. *Comp. Vision Graph.*, 46(1):1–21, 1989.

[5] S.M. Bartlett. An inverse matrix adjustment arising in discriminant analysis. *Ann. Math. Statist.*, 22(1):107–111, 1951.

[6] M. Bazargani, A. Anjos, F.G. Lobo, A. Mollahosseini, and H.R. Shahbazkia. Affine image registration transformation estimation using a real coded genetic algorithm with sbx. *CoRR*, abs/1204.2139, 2012.

[7] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[8] V. Boldea, G.C. Sharp, S.B. Jiang, and D. Sarrut. 4d-ct lung motion estimation with deformable registration: Quantification of motion nonlinearity and hysteresis. *Medical Physics*, 35(3):1008–1018, 2008.

[9] A. Brandt. Multilevel adaptive solutions to BVPs. *Math. Comp.*, 31:333–390, 1977.

[10] W.L Briggs, V.E. Henson, and S.F. McCormick. *A Multigrid Tutorial: Second Edition*. SIAM publications, 2000.

[11] C. Broit. *Optimal Registration of Deformed Images*. PhD thesis, University of Pennsylvania, 1981.

[12] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2009.

[13] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. *ECCV*, 3024:25–36, 2004.

[14] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33:500–513, 2011.

[15] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Real-time optic flow computation with variational methods. *Computer Analysis of Images and Patterns*, 2756:222–229, 2003.

[16] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnörr. Variational optic flow computation in real-time. *IEEE Transactions on Image Processing*, 14:608–615, 2006.

[17] A. Bruhn, J. Weickert, and C. Schnörr. Lucas/kanade meets horn/schunck: combining local and global optic flow methods. *International Journal of Computer Vision*, 61(3):211–231, 2005.

[18] M. Burger, J. Modersitzki, and L. Ruthotto. A hyperelastic regularization energy for image registration. *SIAM Journal on Scientific Computing*, 35(1):B132–B148, 2013.

[19] R.H. Byrd, S.L. Hansen, J. Nocedal, and Y. Singer. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26:1008–1031, 2016.

[20] K. Cao, G.E. Christensen, K. Ding, K. Du, M.L. Raghavan, R.E. Amelon, K.M. Baker, E.A. Hoffman, and J.M. Reinhardt. Tracking regional tissue volume and function change in lung using image registration. *Int. Journal of Biomedical Imaging*, Article ID 956248 (OA), 2012.

[21] E. Castillo, R. Castillo, Y. Zhang, and T. Guerro. Compressible image registration for thoracic computed tomography images. *Journal of Med. Biol. Eng.*, 29(5):222–233, 2009.

[22] G. Cazoulat, D. Owen, M.M. Matuszak ind J. Balter, and K.K. Crock. Biomechanical deformable image registration of longitudinal lung ct images using vessel information. *Physics in Medicine and Biology*, 61:4826–4839, 2016.

[23] K.S.C. Chao, S. Bhide, H. Chen, J. Asper, S. Bush, G. Franklin, V. Kavadi, V. Liengswangwong, W. Gordon, A. Raben, J. Strasser, C. Koprowski, S. Frank,

G. Chronowski, A. Ahamad, R. Malyapa, L. Zhang, and L. Dong. Reduce in variation and improve efficiency of target volume delineation by a computer-assisted system using a deformable image registration approach. *International Journal of Radiation Oncology*Biology*Physics*, 68(5):1512–1521, 2007.

[24] H.M. Chen, M.K. Arora, and P.K. Varshney. Mutual information-based image registration for remote sensing data. *International Journal of Remote Sensing*, 24(18):3701–3706, 2003.

[25] K. Chen. *Matrix Preconditioning Techniques and Applications*. Cambridge University Press, 2005.

[26] Y.M. Chen and X. J. Ye. *Inverse consistent deformable registration*, pages 419–440. Springer, 2010.

[27] G.E. Christensen. *Inverse consistent image registration*, pages 219–250. Springer, 2005.

[28] G.E. Christensen and H.J. Johnson. Consistent image registration. *IEEE Transactions on Medical Imaging*, 20(7):568–582, 2001.

[29] G.E. Christensen, S.C. Joshi, and M.I. Miller. Volumetric transformation of brain anatomy. *IEEE Transactions on Medical Imaging*, 16(6):864–877, 1997.

[30] G.E. Christensen, J.H. Song, W. Lu, I. El Naqa, and D.A. Low. Tracking lung tissue motion and expansion/compression with inverse consistent image registration and spirometry. *Medical Physics*, 34(6):2155–2163, 2007.

[31] N. Chumchob and K. Chen. A robust affine image registration method. *International Journal of Numerical Analysis and Modelling*, 6(2):311–334, 2009.

[32] N. Chumchob and K. Chen. A variational approach for discontinuity preserving image registration. *East-West Journal of Mathematics*, pages 266–282, 2010.

[33] N. Chumchob and K. Chen. A robust multigrid approach for variational image registration models. *Journal of Computational and Applied Mathematics*, 236(5):653–674, 2011.

[34] N. Chumchob and K. Chen. An improved variational image registration model and a fast algorithm for its numerical approximation. *Numerical Methods for Partial Differential Equations*, 28(6):1866–1995, 2012.

[35] N. Chumchob, K. Chen, and C. Brito-Loeza. A fourth order variational image registration model and its fast multigrid algorithm. *Multiscale Moddeling and Simulation*, 9(1):89–128, 2010.

[36] T. Craciunescu, A. Murari, A. Alonso, P.T. Lang, G. Kocsis, and I. Tiseanu. Application of optical flow method for imaging diagnostic in jet. *Journal of Nucelar Materials*, 400:205–212, 2010.

[37] I.J.D. Craig and J.C. Brown. Inverse problems in astronomy: a guide to inversion strategies for remotely sensed data. 1986.

[38] A.R. Cunliffe, C. Contee, S.G. Armato III, B. White, J. Justusson, R. Malik, and H.A. Al-Hallaq. Effect of deformable registration on the dose calculated in radiation therapy planning ct scans of lung cancer patients. *Medical Physics*, 42(1):391–399, 2015.

[39] A.R. Cunliffe, B. White, J. Justusson, C. Straus, R. Malik, H.A. Al-Hallaq, and S.G. Armato. Comparison of two deformable registration algorithms in the presence of radiologic change between serial lung ct scans. *Journal of Digital Imaging*, 28(6):755–760, 2015.

[40] B. Dacorogna. *Direct Methods in the Calculus of Variations*. Springer-Verlag, 1989.

[41] S. Dawn, V. Saxena, and B. Sharma. Remote sensing image registration techniques: a survey. In *Proceedings of the 4th International Conference on Image and Signal Processing*, pages 103–112. Springer-Verlag, 2010.

[42] J. Orban de Xivry, G. Janssens, G. Bosmans, M. De Craene, A. Dekker, J. Buijsen, A. van Baardwijk, D. De Ruysscher, B. Macq, and P. Lambin. Tumour delineation and cumulative dose computation in radiotherapy based on deformable registration of respiratory correlated ct images of lung cancer patients. *Radiotherapy and Oncology*, 85(2):232–238, 2007.

[43] O. Demetz, M. Stoll, S. Volz, J. Weickert, and A. Bruhn. Learning brightness transfer functions for the joint recovery of illumination changes and optical flow. In *ECCV 2014*, pages 455–471, 2014.

[44] K. Ding, J.E. Bayouth, J.M. Buatti, G.E. Christensen, and J.M. Reinhardt. 4dct-based measurement of changes in pulmonary function following a course of radiation therapy. *Medical Physics*, 37(3):1261–1272, 2010.

[45] L.C. Evans. Partial differential equations. 2010.

[46] B. Fischer and J. Modersitzki. Fast inversion of matrices arising in image processing. *Numerical Algorithms*, 22(1):1–11, 1999.

[47] B. Fischer and J. Modersitzki. Fast diffusion registration. *Contemporary Mathematics*, 313:117–128, 2002.

[48] B. Fischer and J. Modersitzki. Curvature based image registration. *Journal of Mathematical Imaging and Vision*, 18(1):81–85, 2004.

[49] B. Fischer and J. Modersitzki. A unified approach to fast image registration and a new curvature based registration technique. *Linear Algebra and its Applications*, 380:107–124, 2004.

[50] B. Fischer and J. Modersitzki. Ill-posed medicine - an introduction to image registration. *Inverse problems*, 24(3), 2008.

[51] C. Frohn-Schauf, S. Henn, L.Hömke, and K. Witsch. Total variation based image registration. In *International Conference on PDE-Based Image Processing and Related Inverse Problems Series: Mathematics and Visualization*, pages 305–323. Springer Verlag, 2006.

[52] C. Frohn-Schauf, S. Henn, and K. Witsch. Multigrid based total variation image registration. *Computing and Visualization in Science*, 11(2):101–113, 2008.

[53] S. Gao, L. Zhang, H. Wang, R. de Crevoisier, D.D. Kuban, R. Mohan, and L. Dong. A deformable image registration method to handle distended rectums in prostate cancer radiotherapy. *Medical Physics*, 33(9):3304–3312, 2006.

[54] J. C. Gee and K. Bajcsy. Elastic matching: continuum mechanical and probabilistic analysis. In *Brain Warping*. Academic Press, 2000.

[55] S. Gefen, O. Tretiak, and J. Nissanov. Elastic 3-d alignment of rat brain histological images. *IEEE Transactions on Medical Imaging*, 22(11):1480–1489, 2003.

[56] D. Gerçek, D. Çeşmeci, M. M.K.G'ull'u, A. Ert'urk, and S. ErtÃ$\frac{1}{4}$rk. Automated co-registration of satellite images through luminance transformation. *The Photogrammetric Record*, 31(156):407–427, 2016.

[57] E. Giusti. *Minimal Surfaces and Functions of Bounded Variation*. Birkhäuser, 1984.

[58] C.K. Glide-Hurst, G.D. Hugo, J. Liang, and D. Yan. A simplified method of four-dimensional dose accumulation using the mean patient density representation. *Medical Physics*, 35(12):5269–5277, 2008.

[59] A.V. Goncharskii, A. Cherepashchuk, and A. Yagola. Ill-posed problems of astrophysics. 1985.

[60] A. Gooya, G. Biros, and C. Davatzikos. Deformable registration of glioma images using em algorithm and diffusion reaction modelling. *IEEE Transactions on Medical Imaging*, 30(2):375–390, 2011.

[61] V. Gorbunova, P. Lo, H. Ashraf, A. Dirksen, M. Nielsen, and M. de Bruijne. *Weight preserving image registration for monitoring disease progression in lung CT*, pages 863–870. Springer Berlin Heidelberg, 2008.

[62] V. Gorbunova, J. Sporring, P. Lo, M. Loeve, H.A. Tiddens, M. Nielsen, A. Dirksen, and M. de Bruijne. Mass preserving image registration for lung CT. *Medical Image Analysis*, 16(4):786–795, 2012.

[63] N.M. Grosland, R. Bafna, and V.A. Magnotta. Automated hexahedral meshing of anatomic structures using deformable registration. *Computer Methods in Biomechanics and Biomedical Engineering*, 12(1):35–43, 2009.

[64] T. Guerrero, G. Zhang, W. Segars, T.C. Huang, S. Bilton, G. Ibbott, L. Dong, K. Forster, and K.P. Lin. Elastic image mapping for 4-d dose estimation in thoracifc radiotherapy. *Radiation Protection Dosimetry*, 115(1-4):497–502, 2005.

[65] T. Guerro, K. Sanders, E. Castillo, Y. Zhang, L. Bidaut, T. Pan, and R. Komaki. Dynamic ventillation imaging from four-dimensional computed tomography. *Phys Med Biol.*, 51(4):777–791, 2006.

[66] C. Guetter, H. Xue, C. Chefd'hotel, and J. Guehring. Efficient symmetric and inverse-consistent deformable registration through inter-leaved optimization. *IEEE International Symposium on Biomedical Imaging From Nano to Macro*, 2011.

[67] E. Haber, S. Heldmann, and J. Modersitzki. Adaptive mesh refinement for non-parametric image registration. *SIAM J. Scientific Computing*, 30:3012–3027, 2008.

[68] E. Haber and J. Modersitzki. A multilevel method for image registration. *SIAM J. Sci. Comput.*, 27(5):1594–1607, 2006.

[69] W.J. Hack. Automated image registration for the future. *Astronomical Data Analysis Software and Systems*, 394:35–44, 2008.

[70] P.M. Harari, S. Song, and W.A. Tomé. Emphasizing conformal avoidance versus target definition for imrt planning in head-and-neck cancer. *International Journal of Radiation Oncology*Biology*Physics*, 77(3):950–958, 2010.

[71] P.W. Hemker. On the order of prolongations and restrictions in multigrid procedures. *JCAM*, 32(3):423–429, 1990.

[72] S. Henn. A multigrid method for a fourth-order diffusion equation with application to image processing. *SIAM Journal on Scientific Computing*, 27(3):831–849, 2005.

[73] S. Henn and K. Witsch. Iterative multigrid regularization techniques for image matching. *SIAM Journal on Scientific Computing*, 23(4):1077–1093, 2001.

[74] S. Henn. and K. Witsch. Image registration based on multiscale energy information. *Multiscale Modeling & Simulation*, 4(2):584–609, 2005.

[75] S. Hermann and R. Werner. High accuracy optical flow for 3d medical image registration using the census cost function. *PSIVT 2013*, LNCS 8333:23–35, 2014.

[76] D.L.G. Hill, P.G. Batchelor, M. Holden, and D.J. Hawkes. Medical image registration. *Physics in medicine and biology*, 46(3):1–45, 2001.

[77] E. Hinton and B. Irons. Least squares smoothing of experimental data using finite elements. *Strain*, 4(3):24–27, 1968.

[78] L. Hömke. A multigrid method for anisotropic pdes in elastic image registration. *Numerical Linear Algebra with Applications*, 13:215–229, 2006.

[79] B.K.P. Horn and B.G.Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[80] G.D. Hugo, E. Weiss, W.C. Sleeman, S. Balik, P.J. Keall, J. Lu, and J.F. Williamson. A longitudinal four-dimensional computed tomography and cone beam computed tomography dataset for image-guided radiation therapy research in lung cancer. *Medical Physics*, 44(2):762–771, 2017.

[81] J. Jiang, S. Cao, G. Zhang, and Y. Yuan. Shape registration for remote-sensing images with background variation. *International Journal of Remote Sensing*, 34(15):5265–5281, 2013.

[82] H.J. Johnson and G.E. Christensen. Consistent landmark and intensity-based image registration. *IEEE Transactions on Medical Imaging*, 21(5):450–461, 2002.

[83] N. Kadoya, S. Kabus, C. Lorenz, M. Diehn, B. Loo, P. Keall, and T. Yamamoto. Radiation dose changes pulmonary function measured by 4d-ct ventilation imaging. *Medical Physics*, 40:520, 2013.

[84] C.T. Kelley. *Iterative methods for optimization*, volume 18. Siam, 1999.

[85] D. Knaan and L. Joskowicz. Effective intensity-based 2d/3d rigid registration between fluoroscopic x-ray and ct. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2003*, pages 351–358, 2003.

[86] H. Köstler, K. Ruhnau, and R. Wienands. Multigrid solution of the optical flow system using a combined diffusion- and curvature-based regularizer. *Numerical Linear Algebra with Applications*, 15(2-3):201–218, 2008.

[87] S. Kullback. Wiley, 1959.

[88] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951.

[89] K. C. Lam and L. M. Lui. Landmark-and intensity-based registration with large deformations via quasi-conformal maps. *SIAM Journal on Imaging Sciences*, 7(4):2364–2392, 2014.

[90] J. Larrey-Ruiz and J. Morales-Sánchez. Optimal parameters selection for non-parametric image registration methods. In *International Conference on Advanced Concepts for Intelligent Vision Systems*, pages 564–575, 2006.

[91] K. Latifi, T.J. Dilling, V. Feygelman, E.G. Moros, C.W. Stevens, J.L. Montylla-Soler, and G.G. Zhang. Impact of dose on lung ventilation change calculated from 4d-ct using deformable image registration in lung cancer patients treated with sbrt. *Journal of Radiation Oncology*, 4(3):265–270, 2015.

[92] Y.T. Lee, K.C. Lam, and L.M. Lui. Landmark-matching transformation with large deformation via n-dimensional quasi-conformal maps. *J. Sci. Comput.*, 67(3):926–954, 2016.

[93] H.S. Li, H. Zhong, J. Kim, C. Glide-Hurst, M. Gulam, T.S. Nurushev, and I.J. Chetty. Direct dose mapping versus energy/mass transfer mapping for 4d dose accumulation: fundamental differences and dosimetric consequences. *Physics in Medicine and Biology*, 59(1):173–188, 2013.

[94] T. Lin, C. Le Guyader, I.D. Dinov, P.M. Thompson, A.W. Toga, and L.A. Vese. Gene expression data to mouse atlas registration using a nonlinear elasticity smoother and landmark points constraints. *J. Sci. Comput.*, 50:586–609, 2012.

[95] H. Livyatan, Z. Yaniv, and L. Joskowicz. Gradient-based 2-d/3-d rigid registration of fluoroscopic x-ray to ct. *IEEE Transactions on Medical Imaging*, 22(11):1395–1406, 2003.

[96] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. volume 7, pages 674–679, 1981.

[97] F.J. Masci, D. Makovoz, and M. Moshir. A robust algorithm for the pointing refinement and registration of astronomical images. *Publications of the Astronomical Society of the Pacific*, 116, 2004.

[98] M. Mathieu and Y. LeCun. Fast approximation of rotations and hessians matrices. *CoRR*, abs/1404.7195, 2014.

[99] A. Meyer-Baese, J. Massich, G. Lemaitre, and M. Rastgoo. Optical flow with theoretically justified warping applied to medical imaging. volume 3, pages 89–95, 2015.

[100] J. Modersitzki. *Numerical Methods for Image Registration*. Oxford University Press, 2004.

[101] J. Modersitzki. *Flexible Algorithms for Image Registration.* SIAM publications, 2009.

[102] M. Moteabbed, G.C. Sharp, Y. Wang, A. Trofimov, J.A. Efstathiou, and H.M. Lu. Validation of a deformable image registration technique for cone beam ct-based dose verification. *Medical Physics*, 42(1):196–205, 2015.

[103] K. Murphy, B. van Ginneken, E. M van Rikxoort, B. J. de Hoop, M. Prokop, P. Lo, M. de Bruijne, and J.P.W. Pluim. Obstructive pulmonary function: patient classification using 3d registration of inspiration and expiration ct images. 2009.

[104] L. Pan, M. Urban, and B. Rolf. Combination of intensity-based image registration with 3d simulation in radiation therapy. *Phys Med Biol.*, 53(17):4621–4637, 2008.

[105] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *International Journal of Computer Vision*, 67:141–158, 2006.

[106] J. Persson, H. Knutsson, and M. Borga. Automatic hip bone segmentation using non-rigid registration. volume 3, pages 946–949, 2006.

[107] A. Pevsner, B. Davis, S. Joshi, A. Hertanto, J. Mechalakos, E. Yorke, K. Rosenzweig, S. Nehmeh, Y.E. Erdi, J.L. Humm, S. Larson, C.C. Ling, and G.S. Mageras. Evaluation of an automated deformable image matching method for quantifying lung motion in respiration-correlated ct images. *Medical Physics*, 33(2):369–376, 2006.

[108] T. Pock, M. Urschler, C. Zach, R. Beichel, and H. Bischof. A duality based algorithm for tv-l^1-optical-flow image registration. *LNCS*, 4792:511–518, 2007.

[109] T. Poggio and V. Torre. Ill-posed problems and regularization analysis in early vision. Technical report, Cambridge, MA, USA, 1984.

[110] V. Reed, W. Woodward, L. Zhang, E.A. Strom, G. Perkins, W. Tereffe, J.L. Oh, T. Yu, I. Bedrosian, G. Whitman, T. Buchholz, and L. Dong. Automatic segmentation of whole breast using atlas approach and deformable image registration. *International Journal of Radiation Oncology, Biology, Physics*, 73:1493–500, 2008.

[111] J.M. Reinhardt, K. Ding, K. Cao, G.E.Christensen, E.A. Hoffman, and S.V. Bodas. Registration-based estimates of local lung tissue expansion compared to xenon ct measures of specific ventilation. *Medical Image Analysis*, 12(6):752–763, 2008.

[112] K. Rektorys. *The biharmonic operator. (equations of plates and wall-beams.)*, pages 266–275. Springer Netherlands, 1977.

[113] M. Reuter, H. Rosas, and B. Fischl. Highly accurate inverse consistent registration: a robust approach. *Neuroimage*, 53 (4):1181–1196, 2010.

[114] E. Rietzel, G.T.Y. Chen, N.C. Choi, and C.G. Willet. Four-dimensional image-based treatment planning: Target volume segmentation and dose calculation in the presence of respiratory motion. *International Journal of Radiation Oncology*Biology*Physics*, 61(5):1535–1550, 2005.

[115] M. Rosu, I.J. Chetty, J.M. Balter, M.L. Kessler, D.L. McShan, and R.K. Ten Haken. Dose reconstruction in deforming lung anatomy: Dose grid size effects and clinical implications. *Medical Physics*, 32(8):2487–2495, 2005.

[116] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259–268, 1992.

[117] L. Ruthotto, C. Greif, and J. Modersitzki. A stabilized multigrid solver for hyperelastic image registration. *Numerical Linear Algebra with Applications*, 24(5), 2017.

[118] Y. Saad. *Intensity based deformable image registration of lung 4DCT*, volume 60.

[119] D. Sarrut, V. Boldea, S. Miguet, and C. Ginestet. Simulation of four-dimensional ct images from deformable registration between inhale and exhale breath-hold ct scans. *Medical Physics*, 33(3):605–617, 2006.

[120] P. Schwind, S. Suri, P. Reinartz, and A. Siebert. Applicability of the sift operator to geometric sar image registration. *International Journal of Remote Sensing*, 31(8):1959–1980, 2010.

[121] S. Shahhosseini, B. Rezaie, and V. Emamian. Sequential image registration for astronomical images. In *2012 IEEE International Symposium on Multimedia*, pages 314–317, 2012.

[122] C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[123] F. Song, L. Deng, G. Shu, F. Wang, H. Deng, and K. Ji. A subpixel registration algorithm for low psnr images. pages 626–630, 2012.

[124] M. Staring, M.E. Bakker, D.P. Shamonin, J. Stolk, J.H.C. Reiber, and B.C. Stoel. Towards local estimation of emphysema progression using image registration. In *SPIE Conference Series*, volume 7529, 2009.

[125] K C. Strasters, J.A. Little, J. Buurman, D.L.G. Hill, and D.J. Hawkes. *Anatomical landmark image registration: Validation and comparison*, pages 161–170. Springer Verlag, 1997.

[126] M. Stürmer and H. Köstler U. Rüde. A fast multigrid solver for applications in image processing. *Numer. Linear Algebra with Appl.*, 15:187–200, 2008.

[127] M. Tahk, M. Park, H. Woo, and H. Kim. Hessian approximation algorithms for hybrid optimization methods. *Engineering Optimization*, 41(7):609–633, 2009.

[128] W.C. Thacker. Oceanographic inverse problems. *Physica D: Nonlinear Phenomena*, 60(1):16–37, 1992.

[129] A.N. Tikhonov and V.Y. Arsenin. *Solutions of ill-posed problems.* Winston ; distributed solely by Halsted Press Washington : New York, 1977.

[130] D. Tomazevic, B. Likar, T. Slivnik, and F. Pernus. 3-d/2-d registration of ct and mr to x-ray images. *IEEE Transactions on Medical Imaging*, 22(11):1407–1416, 2003.

[131] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid.* Academic Press, 2001.

[132] H. Ue, H. Haneishi, H. Iwanaga, and K. Suga. Nonlinear motion correction of respiratory-gated lung spect images. *IEEE Transactions on Medical Imaging*, 25(4):486–495, 2006.

[133] B. Ummenhofer. Large displacement optical flow for volumetric image sequences. In *Pattern Recognition*, Lecture Notes in Computer Science, pages 432–437. Springer Berlin / Heidelberg, 2011.

[134] M. Velec, J.L. Moseley, C.L. Eccles, T. Craig, M.B. Sharpe, L.A. Dawson, and K.K. Brock. Effect of breathing motion on radiotherapy dose accumulation in the abdomen using deformable registration. *International Journal of Radiation Oncology*Biology*Physics*, 80(1):265–272, 2011.

[135] Y. Vinogradskiy, R. Castillo, E. Castillo, S.L. Tucker, Z. Liao, T. Guerrero, and M.K. Martel. Use of 4-dimensional computed tomography-based ventilation imaging to correlate lung dose and function with clinical outcomes. *International Journal of Radiation Oncology*Biology*Physics*, 86(2):366–371, 2013.

[136] H. Wang, L. Dong, J. O'Daniel, R. Mohan, A.S. Garden, K. Kian Ang, D.A. Kuban, M. Bonnen, J.Y. Chang, and R. Cheung. Validation of an accelerated 'demons' algorithm for deformable image registration in radiation therapy. *Physics in Medicine and Biology*, 50(12):2887–2905, 2005.

[137] H. Wang, A.S. Garden, L. Zhang, X. Wei, A. Ahamad, D.A. Kuban, R. Komaki, J. O'Daniel, Y. Zhang, R. Mohan, and L. Dong. Performance evaluation of automatic anatomy segmentation algorithm on repeat or four-dimensional computed tomography images using deformable image registration method. *International Journal of Radiation Oncology*Biology*Physics*, 72(1):210–219, 2008.

[138] Z. Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions On Image Processing*, 13(4):600–612, 2004.

[139] J. Weickert, B. ter Haar Romeny, and M.A. Viergever. Efficient and reliable schemes for nonlinear diffusion filtering. *IEEE Transactions on Image Processing*, pages 398 – 410, 1998.

[140] F. Werner and T. Hohage. Convergence rates in expectation for tikhonov-type regularization of inverse problems with poisson data. *Inverse problems*, 28(10), 2012.

[141] J. Wienands and W. Joppich. *Practical fourier analysis for multigrid method*. Chapman and Hall/CRC, USA, 2005.

[142] P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[143] P. Wolfe. Convergence conditions for ascent methods. ii: some corrections. *SIAM Review*, 13(2):185–188, 1971.

[144] S. Wright and J. Nocedal. Numerical optimization. *Springer Science*, 35:7, 1999.

[145] Z. Wu, E. Rietzel, V. Boldea, D. Sarrut, and G.C. Sharp. Evaluation of deformable registration of patient lung 4dct with subanatomical region segmentations. *Medical Physics*, 35(2):775–781, 2008.

[146] T. Yamamoto, S. Kabus, C. Lorenz, E. Johnston, P.G. Maxim, M. Diehn, N. Eclov, C. Barquero, B.W. Loo, and P.J. Keall. 4d ct lung ventilation images are affected by the 4d ct sorting method. *Medical Physics*, 40(10), 2013.

[147] D. Yang, H. Li, D.A. Low, J.O. Deasy, and I. El Naqa. A fast inverse consistent deformable image registration method based on symmetric optical flow computation. *Physics in Medicine & Biology*, 53(21):6143, 2008.

[148] L.J. Yang, Z. Tian, and W. Zhao. A new affine invariant feature extraction method for sar image registration. *International Journal of Remote Sensing*, 35(20):7219–7229, 2014.

[149] I. Yanovsky, P. M. Thompson, S. Osher, and A. D. Leow. Topology preserving log-unbiased nonlinear image registration: theory and implementation. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[150] I. Yanovskyr, C. Le Guyader, A. Leow, P. Thompson, and L. Vese. Unbiased volumetric registration via nonlinear elastic regularization. 2008.

[151] Y. Yin, E.A. Hoffman, and C.L. Lin. Local tissue-weight-based nonrigid registration of lung images with application to regional ventilation, 2009.

[152] Y. Yin, E.A. Hoffman, and C.L. Lin. Mass preserving nonrigid registration of ct lung images using cubic b-spline. *Medical Physics*, 36(9):4213–4222, 2009.

[153] D. P. Zhang and K. Chen. A novel diffeomorphic model for image registration and its algorithm. *Journal Of Mathematical Imaging And Vision*, DOI: 10.1007/s10851-018-0811-3, 2018.

[154] P. Zhang, G.D. Hugo, and D. Yan. Planning study comparison of real-time target tracking and four-dimensional inverse planning for managing patient respiratory motion. *International Journal of Radiation Oncology*Biology*Physics*, 72(4):1221–1227, 2008.

[155] L. Zhao, A.K. Venkatesh, G.P Zientara, F.A. Jolesz, M.S Albert, and L.P Panych. Optical flow of rat lung dynamics using hyperpolarized noble gas mri. *Proc. Intl. Soc. Mag. Reson. Med.*, 8:2188, 2000.

[156] D. Zikic, W. Wein, A. Khamene, D.A. Clevert, and N. Navab. *Fast deformable registration of 3D-ultrasound data using a variational approach*, pages 915–923. Springer Berlin Heidelberg, 2006.